Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

CRT-2006-22

MODELS AND TABU SEARCH META-HEURISTICS FOR SERVICE NETWORK DESIGN WITH VEHICLE BALANCE REQUIREMENTS

by

Michael Berliner Pedersen
Teodor Gabriel Crainic
Oli B.G. Madsen

November 2006

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone: (514) 343-7575 / Télécopieur: (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

HEC ÉCOLE POLYTECHNIQUE MONTRÉAL Université de Montréal

2007-00274

PUBLICATION
CRT-2006-22

MODELS AND TABU SEARCH META-HEURISTICS FOR SERVICE NETWORK DESIGN WITH VEHICLE BALANCE REQUIREMENTS

by

Michael Berliner Pedersen [1]
Teodor Gabriel Crainic [2]
Oli B.G. Madsen [1]

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)

1. Centre for Traffic and Transport, Technical University of Denmark, Anker Engelundsvej 1, Building 101 A, 2800 kgs., Lyngby, Denmark (mbp, ogm@ctt.dtu.dk)
2. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 and Département de management et technologie, École des sciences de la gestion, Université du Québec à Montréal, C.P. 8888, succursale Centre-ville, Montréal, Canada H3C 3P8 (theo@crt.umontreal.ca)

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2006

Abstract

In this paper we present a generic model for solving service network design problems in transportation with additional vehicle balance constraints. The motivation for the model comes from the possibility of reducing empty vehicle flows within transportation systems by addressing the conservation of vehicle/equipment flow when scheduling services. The model is an extension of the fixed-charge capacitated multi-commodity network design model (CMND). The extension consists of adding a new set of constraints denoted design balance constraints resulting in the design balanced capacitated multi-commodity network design model (DBCMND). The constraints require that the number of selected design arcs entering a node be equal to the number of selected design arcs leaving a node. The paper furthermore presents a Tabu Search heuristic framework for solving the arc-based formulation of the DBCMND. The performance of the algorithm is measured against the performance of the Xpress-MP optimization suite MIP-solver.

Keywords: Service network design, fixed cost, multicommodity capacitated network design, balance constraints, tabu search meta-heuristics

Résumé

Nous présentons un modèle pour le problème de design de réseau de service avec des restrictions supplémentaires relativement à la circulation des véhicules. La motivation pour ce modèle provient de la possibilité d'améliorer l'utilisation de la flotte de véhicules en intégrant les considérations de conservation des flux d'équipement aux nœuds/terminaux du réseau lorsque le plan-horaire des services est conçu. Le modèle est une extension de la formulation de design de réseaux multiproduits avec capacités et coûts fixes, par l'ajout de contraintes de balance de design imposant que le nombre d'arcs de design sélectionnés entrant dans un nœud soit égal à celui des arcs de design sélectionnés sortant du nœud. L'article propose une méta-heuristique basée sur les recherches avec tabous dont l'efficacité est mesurée en comparant ses résultats à ceux d'Express-MP sur un grand ensemble de problèmes test.

Mots-clés: design de réseaux de service, design de réseaux multiproduits avec capacités et coûts fixes, contraintes de balancement, méta-heuristiques de recherche avec tabous.

1. Introduction

Network design models represent generic models for a wide range of applications in planning transportation, logistics, telecommunication, and production systems. In these applications, multiple commodities (goods, data, people, etc.) must be routed between different points of origin and destination over a network of nodes and arcs with possibly limited capacity. Moreover, other than the routing cost proportional to the number of units of each commodity transported over a network link, a fixed cost must be paid the first time the link is used, representing its construction (opening), improvement costs, or service functioning cost. The general network design problem consists of finding a minimum cost design i.e. a choice of arcs in the network to enable the flow of commodities such that it minimizes the sum of the fixed cost of including the arcs and the variable cost of routing the commodities on them. Presentation of different network design models and their applications can be found in Minoux (1986), Magnanti and Wong (1984), Ahuja et al. (1995, and Balakrishnan et al. (1997)).

Service network design is an application within transportation systems where one or more issues, such as commodity consolidation, service type choice, service frequency, delivery time windows, terminal congestion, schedules, and equipment positioning may be considered. The planning scope is generally on a tactical level, as opposed to the strategic scope of network design models. The service network design problem for freight transportation is described in Crainic (2000) and applications can be found in Barnhart and Kim (1995), Cheung and Muralidharan (2000), and Powell and Sheffi (1989) for road transportation, Joborn et al. (2004), Marin and Salmeron (1996), Newman and Yano (2000), and Cordeau et al. (1998) for train transportation, Kuby and Gray al. (1993) for air transportation, and Armacost et al. (2002), Kim et al. (1999), Nozick and Morlok (1997), and Jansen et al. (2004) for various intermodal transportation problems. Although some effort has been put into vehicle balancing and repositioning (Dejax and Crainic (1987)) presents a survey on the issue) it is generally implicitly assumed in service network design models that equipment is available when needed. The fleet assignment, vehicle positioning, and empty vehicle repositioning problems are dealt with at an operational level on an already decided service network schedule.

With carriers operating with minimal fleet sizes, the vehicle positioning can significantly impact the choice of services offered. An example is the rotations that intercontinental ships perform. Each leg between calling ports can be considered a service. The individual legs are interconnected by the fact that they can not be offered unless the preceding leg on the rotation is offered (i.e. a ship can only leave port if it has previously arrived). The resulting service network is a series of services following each other resulting in rotations of services for vehicles. There are several other examples of service network schedules operating according to a cyclic service rotation over a period of time. These are e.g. within public transportation, airline routes, and train routes. Little effort has been dedicated to designing service network schedules whilst simultaneously considering vehicle positioning and balancing.

The contribution of this paper is twofold. First the paper presents a model for service network design which includes equipment positioning and utilization. This is addressed by adding additional constraints on the requirement of equipment availability in order to perform services. As will be shown, the model can be considered an extension of the capacitated multi-commodity network design model (CMND) with a set of, what we call "design balance constraints", added. We denote the resulting model the design balanced capacitated multi-commodity network design model (DBCMND) and present both an arc-based and a cycle-cased formulation of the model. Second the paper proposes a Tabu Search heuristic

framework to solve the arc-based formulation of the DBCMND. The algorithm is applied on selected network design instances from Crainic et al. (2000) and impressive computational results are presented.

2. Problem modelling

The vehicle balanced scheduled service network design problem can be stated as: Select scheduled services, such that all commodities can be routed from their origin to their destination within their specified time window, while ensuring that the availability of vehicles for performing the services is maintained in the terminals.

We introduce a set of nodes $\mathcal{N}$ representing a terminal at a given time interval. Figure 1 show three terminals each represented by seven nodes each representing a time period of one day.

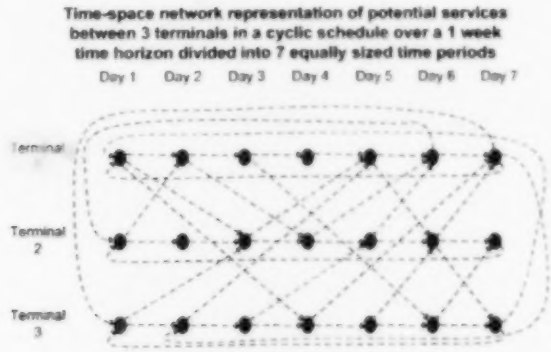
Time-space network representation of potential services between 3 terminals in a cyclic schedule over a 1 week time horizon divided into 7 equally sized time periods
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Terminal
Terminal 2
Terminal 3

FIGURE 1.

We introduce a set of services $\mathcal{A}$ that represent the set of scheduled services between terminals that may be included in the service network. Assuming all services depart from terminals at the end of time periods and arrive at the beginning of time periods we can represent a service as an arc between two nodes $(i,j)|i,j \in \mathcal{N}$. All arcs $(i,j) \in \mathcal{A}$ are, without loss of generality, considered to be design arcs. Figure 1 illustrates a number of potential services (non-horizontal lines between nodes) e.g. with a possible service departing from the node representing terminal 3 at the end of day 1 that arrives at the node representing terminal 2 at the beginning of day 3.

We introduce a set of commodities $\mathcal{P}$ that represent possibly various physical commodities with various origin and destination terminals, and a time of availability and delivery time. To model this, each commodity $p \in \mathcal{P}$ is associated with an origin node $o(p)$ and a destination node $s(p)$ thus capturing both the geographic and temporal attributes of the commodity. To capture the commodity flow requirement of w_p units, a continuous variable x_{ij}^p, $(i,j) \in \mathcal{A}, p \in \mathcal{P}$, is associated to each service arc for each commodity. The operational costs incurred from commodities on a service (e.g. variable fuel costs, handling cost etc.), are modelled by a unit cost c_{ij}^p associated to each service arc for each commodity.

In order not to complicate the model unnecessarily a number of assumptions have been made. It is assumed that all services in $\mathcal{A}$ are performed by the same vehicle type. A vehicle type may be truck, ship, train, bus etc. It is also assumed that only one vehicle is used to perform a service. This is reasonable for single entity vehicles such as busses or ships, but less apparent e.g. for rail. However, as seen in Europe,

intermodal shuttle trains and passenger trains often consist of a fixed configuration and can therefore be considered as an entity. Furthermore, addressing maintenance constraints and/or equipment reconfiguration decisions (e.g. train make-up changes) explicitly, is excluded. Such possible extensions are left to future research.

We associate a decision variable y_{ij} that is 1, if arc (i,j) is opened representing that the service is performed and vehicle is assigned to it, and 0 else. Performing a service has a number of commodity independent operational costs associated to it (e.g. right of passage, infrastructure cost, vehicle depreciation cost or leasing cost etc). To capture the costs we use a fixed cost f_{ij} for each service $(i,j)\in \mathcal{A}$ representing the total cost of performing the service. It cannot be assumed that capacity is the same on each service. Volume, weight, and/or speed restrictions on services, given from physical infrastructure limitations or legislative issues, may affect the capacity offered on a service. Each service arc has a limited capacity u_{ij} for routing commodities.

Note that it is assumed the designed service network is repetitive over a given time horizon (in Figure 1 seven days). Schedule repetition is seen within several types of transportation service networks (e.g. urban public transit, airline systems, European intermodal train operations, intercontinental liner shipping etc). The repetitiveness is in Figure 1 represented by arcs going "backwards" in time. The "backwards" arcs represent services in operation at the transition from one schedule repetition to the next at the time horizon. E.g. the arc going from terminal 1 at the end of day 6 to terminal 2 at the beginning of day 1 is a "backward" arc representing a service leaving on day 6 in one week and arriving on day 2 of the following week.

In figure 1 there are a horizontal arcs connecting two nodes representing the same terminal at different time periods. These are referred to as holding arcs. Holding arcs do not represent a service but a possibility for vehicles staying at a terminal from one time period to another. As for services, letting the vehicle stay at a terminal also incurs costs (such as terminal parking, leasing cost, depreciation value etc.). Thus letting a vehicle stay at a terminal is also from a modelling point of view, a necessary decision to be made. The holding arcs are here included as "pseudo services" in the set of service arcs $\mathcal{A}$ and we shall not distinguish between holding arcs and service arcs but just consider them as temporal connections between two terminal nodes with a fixed cost associated to them.

The novelty introduced in the problem formulation is addressing vehicle balance requirements in the terminals. Consider the following. Each service is performed by one vehicle, eventually resulting in a number of vehicles leaving and arriving terminal nodes. In order to achieve a vehicle balance in terminal nodes the number of services destined to a node must equal the number of services emanating from the same node. Otherwise, the number of vehicles arriving does not match the number departing at a terminal within a time period and an imbalance of vehicles will occur. Adding balance requirement in each node ensuring this relation addresses the vehicle availability and positioning implicitly.

The following section presents a mathematical formulation to the problem modelled above.

3. The Design Balanced Service Network Design model

Recall that $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ the set of directed arcs (services). Recall further that all $(i,j)\in \mathcal{A}$, $i,j\in \mathcal{N}$, are design arcs. Let us further define:

- $\mathcal{N}^+(i):\{j\in \mathcal{N}^+ \mid (i,j)\in \mathcal{A}\}$ as the outward neighbours of node i.

- $\mathcal{N}^-(i):\{j\in\mathcal{N}^-\,|\,(j,i)\in\mathcal{A}\}$, as the inward neighbours of node i.

Recall that for each arc y_{ij}, $(i,j)\in\mathcal{A}$, is representing the design variable that is equal 1 if arc (i,j) is selected in the solution (and 0 otherwise). Similarly x_{ij}^p, $(i,j)\in\mathcal{A}, p\in\mathcal{P}$, represents the flow distribution decision variables indicating the amount of flow of commodity p on arc (i,j). Let X and Y represent the vectors of flow and design variables respectively. $\mathcal{P}$ denotes the set of commodities to move using this network, where each commodity p has a single origin $o(p)$, a single destination $s(p)$, and a flow requirement of w_p units between its origin and destination nodes. For each node i let:

$$d_i^p=\begin{cases} w^p & \text{if } i=o(p)\\ -w^p & \text{if } i=s(p)\\ 0 & \text{otherwise}\end{cases}$$

Finally, the following parameters are defined:

- u_{ij}: capacity of arc (i,j)
- f_{ij}: fixed cost applied on arc (i,j)
- c_{ij}^p: cost of one unit flow of commodity p on arc (i,j)

The arc-based formulation of the vehicle balanced scheduled service network design problem can then be written as follows:

$$\min\ z(X,Y)=\sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij}+\sum_{p\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}} c_{ij}^p x_{ij}^p \qquad (1)$$

$$\sum_{j\in\mathcal{N}^+(i)} x_{ij}^p-\sum_{j\in\mathcal{N}^-(i)} x_{ji}^p=d_i^p,\quad \forall i\in\mathcal{N},\forall p\in\mathcal{P} \qquad (2)$$

$$\sum_{j\in\mathcal{N}^-(i)} y_{ji}-\sum_{j\in\mathcal{N}^+(i)} y_{ij}=0\quad \forall i\in\mathcal{N} \qquad (3)$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p\le u_{ij}y_{ij},\quad \forall(i,j)\in\mathcal{A} \qquad (4)$$

$$x_{ij}^p\ge 0,\quad \forall(i,j)\in\mathcal{A},\quad \forall p\in\mathcal{P} \qquad (5)$$

$$y_{ij}\in\{0,1\},\quad \forall(i,j)\in\mathcal{A} \qquad (6)$$

The objective function (1) accounts for the total system cost, the fixed cost of arcs included in a given design plus the cost of routing the commodity demand, and aims to select the minimum cost design. Constraints (2) represent the network flow conservation relations for commodities. Constraints (3) are the design balance constraints. The constraints state that the total number of open design arcs going into a node i must equal the number of open design arcs going out of node i. Finally, constraints (4) state that for each arc, the total flow of all commodities cannot exceed its capacity if the arc is opened ($y_{ij}=1$) and must be 0 if the arc remains closed ($y_{ij}=0$), while relations (5) and (6) are the usual non-negativity and integrality constraints for the decision variables.

Notice that by removing constraints (3) from the formulation, the general arc-based capacitated multi-commodity network design model (CMND) formulation is obtained. We thus dub the model the design balanced capacitated network design model (DBCMND). Further notice that, as for the CMND formulation, the capacitated multi commodity minimum cost flow problem (CMCF) is obtained for a given fixed design vector, $\bar{Y}: \bar{y}_{ij} = 1$:

$$\min\ z(x(\bar{Y})) = \sum_{p \in \mathcal{P}} \sum_{(i,j) \in \mathcal{A}(\bar{Y})} c_{ij}^p x_{ij}^p \qquad (7)$$

$$\sum_{j \in \mathcal{N}^+(i)} x_{ij}^p - \sum_{j \in \mathcal{N}^-(i)} x_{ji}^p = d_i^p \quad \forall i \in \mathcal{N}, \forall p \in \mathcal{P} \qquad (8)$$

$$\sum_{p \in \mathcal{P}} x_{ij}^p \le u_{ij} \bar{y}_{ij} \quad \forall (i,j) \in \mathcal{A}(\bar{Y}) \qquad (9)$$

$$x_{ij}^p \ge 0 \quad \forall (i,j) \in \mathcal{A}(\bar{Y}), \forall p \in \mathcal{P} \qquad (10)$$

where $\mathcal{A}(\bar{Y}) \subseteq \mathcal{A}$ stands for the set of arcs corresponding to the design $\bar{Y}$.

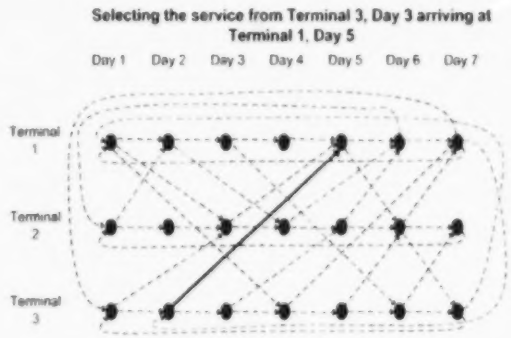

FIGURE 2.

The design balance constraints (3) in the a-DBCMND create further interdependency between the arc choices, in addition to the commodity flow and capacity constraints in the CMND formulation. The effect of the additional interdependency can be illustrated by the following consideration. In figure 2 an arc from the network in figure 1 is opened. The two nodes connected by the selected service (Terminal 3, day 2 and Terminal 1, day 5) are now "out of balance" by not obeying constraints (3). The balance in the nodes can be restored if additional services are opened as shown in figure 3.

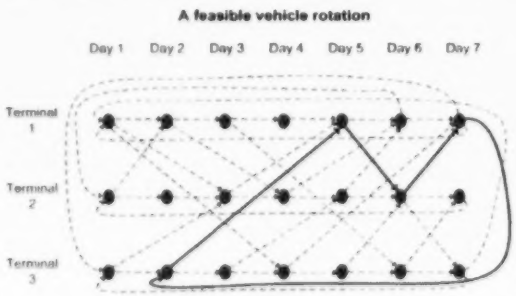

FIGURE 3.

The design in figure 3 is feasible with respect to the design balance constraints (3). Notice how the four selected arcs form a cycle. We formally define a cycle as a set of arcs obeying constraints (3). Recalling that a vehicle is used with each opened arc, a cycle represents a vehicle rotation over the time horizon of a schedule's repetition. By using the notion of cycles we can reformulate the a-DBCMND model. We define the set of cycles $\mathcal{K}$ and the cycle design variables $\eta_k, k \in \mathcal{K}$ that are equal 1 if cycle k is included (and 0 otherwise). Furthermore we define the parameters $a_{ij}^k, (i,j) \in \mathcal{A}, k \in \mathcal{K}$ that are 1 if arc *(i,j)* is in cycle k (0 if not). Using the remaining prior definitions, the a-DBCMND model can be reformulated to obtain the cycle-based formulation, c-DBCMND:

$$\min z(X,Y) = \sum_{(i,j)\in\mathcal{A}} \sum_{k\in\mathcal{K}} f_{ij} a_{ij}^k \eta^k + \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^p x_{ij}^p \quad (11)$$

$$\sum_{j\in\mathcal{N}^+(i)} x_{ij}^p - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^p = d_i^p \quad \forall i \in \mathcal{N}, \forall p \in \mathcal{P} \quad (12)$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \le u_{ij} a_{ij}^k \eta^k \quad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K} \quad (13)$$

$$\sum_{k\in\mathcal{K}} \eta^k a_{ij}^k \le 1 \quad \forall (i,j) \in \mathcal{A} \quad (14)$$

$$x_{ij}^p \ge 0 \quad \forall (i,j) \in \mathcal{A}, \forall p \in \mathcal{P} \quad (15)$$

$$\eta^k \in \{0,1\} \quad \forall k \in \mathcal{K} \quad (16)$$

The capacity constraints (4) have been replaced by constraints (13), that are a set of constraints binding the new cycle-based design decision variables to the arc-based flow variables. The design balance constraints (3) are captured partly by the new cycle-based decision variables definition and partly by constraints (14), that are set-packing constraints stating, that an arc may only be included in one open cycle.

The advantage of the c-DBCMND compared to the a-DBCMND formulation is that the node balance constraints needn't be addressed explicitly in the modelling constraints but are taken care of implicitly in the variable definition and constraints (14). The disadvantage of the c-DBCMND formulation is the size of the set of cycles $\mathcal{K}$. It is computationally infeasible to enumerate all design variables and binding constraints (13) for large scale instances. We will not address the c-DBCMND in this paper but have introduced it for a more complete description of the model, and the insight the cycle-variables give of feasible solution structures.

As previously shown in Magnanti and Wong (1986) un-capacitated fixed-charge network design models are difficult to solve as they belong to the class of NP-hard problems. The capacitated CMND model is even harder (Balakrishnan et al. (1997)) due to, among other factors, the competition of commodities for the network capacity and the difficulty of representing trade-offs between arcs' fixed costs and capacities. Although this will not be formally proved in this paper, adding the node balance constraints intuitively adds further complexity to the problem because of the additional design interdependency of arc choices. By relaxing the design balance constraints (3) in the a-DBCMND formulation, the CMND formulation is obtained which in turn is known to be NP-hard. Furthermore, finding a feasible solution for the CMND problem is trivial. By setting all binary variables to one (opening all arcs) and solving the CMCF problem a feasible solution is obtained. This is not true for the DBCMND problem, where the design balance

constraints prevent the trivial "all arcs open" solution to be feasible, except for special instances. We therefore assume that the DBCMND model belongs to the class of NP-hard problems.

	Section
Initialize:	
Solve the LP-relaxation of the DBCMND model	(4.1)
Round the design variables up to obtain an infeasible starting solution	
Local Search Phase 1 (Exploration phase):	
Establish candidate list of arcs to add or drop $\mathcal{C}_1^{n-1}$ for incumbent solution $n-1$, $\mathcal{C}_1^{n-1} \cap \mathcal{T} = \emptyset$	(4.5)
Evaluate approximate solution value $\bar{V}_c$ for each potential neighbor solution c in $\mathcal{C}_1^{n-1}$	(4.4+4.5)
Select as neighbor solution n the candidate solution c^* with minimum estimated system value, $\bar{V}_{c^*} = \min_{c \in \mathcal{C}_1^{n-1}} \bar{V}_c$	
Update design vector $Y_{n-1} \rightarrow Y_n$ and solve CMCF to get flow vector X_n and calculate V_n	
Update Tabu list, $\mathcal{T}$	(4.6)
Set neighbor solution n to be new incumbent solution $n-1$	
If progress in best found solution is less than I_g% over the past I_l iterations → Local Search Phase 2	(4.2)
Local Search Phase 2 (feasibility phase):	
Establish candidate list of patn to add or drop $\mathcal{C}_2^{n-1}$ for incumbent solution $n-1$	(4.9)
Evaluate solution value V_c for each potential neighbor solution c in $\mathcal{C}_2^{n-1}$ by solving the CMCF	(4.8)
Select as neighbor solution n the candidate solution c^* with minimum $V_{c^*} = \min_{c \in \mathcal{C}_2^{n-1}} V_c$	
Update design vector $Y_{n-1} \rightarrow Y_n$ and set $V_n = V_c$	
Repeat phase 2 search until feasible solution is found or search fails, return to phase 1	(4.9)
Update:	
Keep track of best feasible solution found	
Stop when time limit is reached	(4.1)

Figure 4. Overall algorithmic Structure

It is possible that the particular cycle-structure of design arcs in the feasible solutions for the DBCMND problem may, through further investigation, show that the use of exact methods is applicable to the problem. However, previous attempts to solve the CMND model have indicated that tailored heuristics present the best results for large-scale instances of the generalized network design problem. The focus of this paper is to make an initial attempt at finding good feasible solutions. Therefore, a Tabu Search based heuristic algorithm applied to the arc based formulation of the DBCMND model formulation is used. The algorithm is presented in the following section.

4. Tabu Search heuristic

Tabu Search (TS) belongs to the family of heuristic search methods known as *metaheuristics*. It was introduced in Glover (1986) and has been widely used since (for a survey see Glover and Laguna (1997)). It starts from an initial solution and iteratively moves to a new solution by selecting it from a *solution neighbourhood* of the current *incumbent* solution. The neighbourhood of an incumbent solution is a set of other solutions that can be obtained from the incumbent solution with one or a few simple changes. The transition from the incumbent solution to a neighbour solution is called a *neighbourhood move*. The best solution of the neighbourhood is selected as the new incumbent solution. As mentioned previously the TS

algorithm is based on the arc formulation. The neighbour solutions are here obtained by adding or dropping arcs to the design vector.

TS may be divided into different *phases* each with their own distinctive neighbourhood. The TS framework we present here is composed of two local search phases. The first phase is what could be called an *exploration phase* with the intension to search the solution space by making relatively simple neighbourhood moves between possibly infeasible solutions based on an add/drop procedure on individual arcs. To control infeasibility a penalty value is added to each solution to estimate how far from feasibility it is. The purpose of the penalty value is to have a trade off between cost and infeasibility. The second phase is what could be called a *feasibility phase*. The neighbourhood moves in the second phase are based on an add/drop procedure of paths of design arcs and is intended to find feasible solutions.

The size of a neighbourhood may be very large. It can therefore be computationally demanding to make an exhaustive search of the neighbourhood. To decrease the computational requirements for determining a neighbour move a *candidate list* of potential neighbours to be examined is determined. The candidate list contains a sub-space of potential neighbour solutions that by one, or several, criteria are deemed worth investigating. For both local search phases in the TS algorithm presented here only a subspace of the neighbourhood of an incumbent solution is searched by establishing a candidate list of potential moves.

To avoid the occurrence of cycling between solutions resulting from the movement back and forth to a local optimum, a *tabu list* is introduced. A tabu list is a short-term memory mechanism which stores *attributes* identifying the moves that produced recent solutions. During examination of an incumbent solution's neighbourhood, all the moves that have attributes equal to those stored in the tabu list are discarded. This prevents the search from cycling between two or more solutions.

The time required by each iteration depends on the size of the neighbourhood that needs to be examined, of the time needed to generate each neighbour solution, and the time needed to evaluate each *neighbour solution value*. TS can be stopped after a number of iterations without notable improvements or after a predefined elapse of time. In order to compare the algorithm presented here to a MIP-solver we have chosen to terminate it after a predefined amount of time, τ_{max}. The following subsections present the various elements of the algorithm, whose overall lay-out with section pointers can be seen in figure 4.

4.1. Initialization and starting solution and stopping criterion

In order to start the algorithm an initial solution is necessary. Since the exploration search phase allows moves between infeasible solutions the initial solution need not be feasible. We choose to relax the binary constraints of the a-DBCMND formulation to get the r-DBCMND problem:

$$\min z(X,Y^0) = \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij}^0 + \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^p x_{ij}^p \quad (17)$$

$$\sum_{j\in\mathcal{N}^+(i)} x_{ij}^p - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^p = d_i^p, \quad \forall i\in\mathcal{N}, \forall p\in\mathcal{P} \quad (18)$$

$$\sum_{j\in\mathcal{N}^-(i)} y_{ji}^0 - \sum_{j\in\mathcal{N}^+(i)} y_{ij}^0 = 0, \quad \forall i\in\mathcal{N} \quad (19)$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \le u_{ij} y_{ij}, \quad \forall (i,j)\in\mathcal{A} \quad (20)$$

$$x_{ij}^p \ge 0, \quad \forall (i,j)\in\mathcal{A}, \forall p\in\mathcal{P} \quad (21)$$

$$y_{ij}^0 \in [0,1], \quad \forall (i,j)\in\mathcal{A} \quad (22)$$

Solving the linear programming model given by the r-DBCMND will result in a solution with fractional values on the design variables. By rounding up the fractional values of the design variables, $y_{ij} = 1 \mid y_{ij}^0 > 0 \; \forall (i,j)\in\mathcal{A}$ we get a presumably infeasible starting solution that is used as a starting solution for the neighbourhood search.

4.2. Neighbourhood structure of the exploration phase

An intuitive scheme to adopt for moves between neighbour solutions is the add/drop procedure. Neighbour solutions are obtained by either adding or dropping an arc from the design vector. This approach was adopted in Powell (1986), Koskosidis et al. (1992) and Crainic et al. (1993). The add/drop procedure has proven to be only little effective in solving the general network design problem. This is due to the fact that a single arc is only one of many in a path from an origin to a destination of a commodity and its impact is generally limited. One may often reroute traffic and obtain an almost equivalent solution. Or, as for arc *(a,b)* in Figure 5, it may not even be connected to the other currently open arcs. Introducing such an arc into the network has no influence whatsoever or very little on the flow of the current solution but adds the fixed cost of it to the total system cost. Alternatively, removing an arc may result in an infeasible solution, because connectivity is removed. If e.g. arc *(d,c)* is removed from the solution in the network in figure 5 the commodities with destination node *c* cannot attain their destination, thus rendering the solution infeasible.

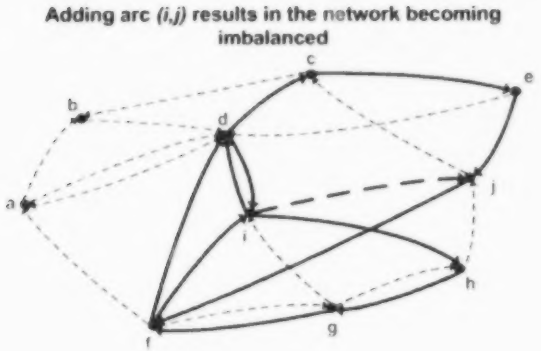

FIGURE 5.

Because of the design balance constraints in the DBCMND model the add/drop procedure has a bigger impact on the solution. Recalling the cycle structure of feasible solutions, adding an arc or dropping one arc from a cycle from will result in an infeasible solution with respect to the design balance constraints. If arc *(i,j)* in Figure 5 is added the nodes *i* and *j* become imbalanced. If in contrary arc *(i,j)* in figure 5 is open, dropping it will achieve feasibility with respect to the design balance constraints. Because of the stronger interconnection between the decision variables the add/drop procedure provides an interesting possibility for a neighbourhood move, and is used as a template for a neighbour move in the exploration phase. We shall show later how the drop moves are enhanced by cycle-based arc opening moves to guarantee commodity flow feasibility.

We will limit the number of arcs added or dropped to one per neighbourhood move. Not considering the eventual cycle-based arc opening enhancement, the total number of potential neighbour solution is thus equivalent the size of the set of arcs, $|\mathcal{A}|$, as each arcs can be closed if open, and opened if closed.

To determine when to stop the exploration phase and proceed to the feasibility phase, two empirical parameters, the *improvement range*, l_i, and the *improvement gap*, l_g are introduced. The improvement range is the number of iterations over which the improvement on the best global total solution value is registered. The improvement gap is the percentage by which the best global total solution value will be required to improve over the improvement range in order to qualify as a significant improvement. Thus if the best global solution value has not improved with l_g% over the last l_i iterations the algorithm switches to the feasibility phase.

4.3. Controlling infeasibility in the exploration phase

Using the add/drop procedure with the cycle-based enhancement does not guarantee feasibility with respect to the design balance constraints. To guide the algorithm's search for feasible neighbour solutions, or neighbour solutions close to feasibility, we implement a scheme to monitor the infeasibility. A *penalty value, P,* is introduced. The penalty value is a pseudo cost that represents an estimate of how far the solution is from feasibility. The penalty value is added to the total system cost, *Z*, giving a *solution value, V,* for a neighbour solution:

$$V = Z + P$$

The solution value is the measure by which neighbour solutions are compared. By the solution value measure a neighbour solution with a relatively high total system cost that is close to feasibility may be better than one with a lower total system cost that is further from feasibility.

The penalty value is obtained using two parameters, the *total system imbalance* and the *maximum absolute imbalance*. Both parameters are calculated from the *node imbalances* in the network. The node imbalance of node i, ψ^i, is calculated as the difference between the number of open outgoing arcs and open ingoing arcs in node i.

$$\psi^i = \sum_{j \in \Omega^+(i)} y_{ji} - \sum_{j \in \Omega^-(i)} y_{ij}$$

Hence the node imbalance is negative if there are more outgoing arcs than ingoing arcs and vice versa. Figure 6 shows two nodes; one with negative imbalance, and one with positive imbalance.

Illustration of node imbalances

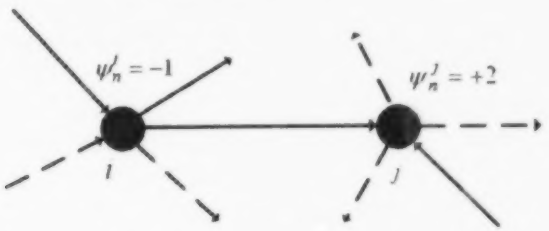

FIGURE 6

The *total system imbalance*, ψ^{Ω}, is calculated as the sum over all absolute node imbalances, whilst the *maximum absolute imbalance*, $\psi^{\max}$, is the largest absolute value of all node imbalances:

$$\psi^{\Omega} = \sum_i |\psi^i|$$

$$\psi^{\max} = \max_i \left(|\psi^i| \right)$$

The total system imbalance gives an indication of the infeasibility of the total system by representing the total number of imbalances that need to be fixed in all the nodes. The maximum node imbalance gives an indication of the difficulty of achieving feasibility in the single node with the highest imbalance.

Using the analogy of scheduled transportation networks, a node imbalance can be interpreted as having to add $|\psi^i|$ empty vehicle movements in or out of node i in order to eliminate the vehicle imbalance. The cost of adding an empty vehicle move, $\bar{f}$, is here approximated as the product of the average of the network arcs' fixed cost, $\tilde{f}$, and an empirical scaling parameter, ψ^p. The empirical parameter is used to externally control the importance of the penalty value in the solution value of a neighbour solution.

By multiplying the cost with the total system imbalance and the maximum node imbalance an estimated value for making a solution feasible is obtained. The penalty value is thus computed as:

$$P = \bar{f} \cdot \psi^{\Omega} \cdot \psi^{\max}, \quad \bar{f} = \tilde{f} \cdot \psi^p$$

In the current implementation the scaling parameter ψ^p is an fixed value that does not change during the progress of the algorithm.

4.4. Neighbourhood move and total system cost evaluation for the exploration phase

Changing the incumbent design vector Y_{n-1}, by adding or dropping an arc, results in a new design vector Y_n which eventually may lead to a new flow vector X_n.

Recall that, given a design vector, the flow vector can be computed by solving the capacitated minimum cost flow problem (CMCF). However, even if the CMCF problem is a LP-problem, solving an instance for each potential neighbour solution is not computationally feasible. Furthermore recall also that closing an arc may lead to losing connectivity and an infeasible solution to the CMCF problem.

To permit fast evaluation and guarantee feasible flow solutions when changing the design for the different potential neighbour solutions, we propose a hybrid of the simple add/drop procedure and a procedure resembling the cycle-based neighbourhoods proposed by Ghamlouche et al. (2003). Cycle-based neighbourhood moves redirect flow around cycles by closing and opening design arcs accordingly. The idea behind the move comes from the acknowledgement that commodities move on paths and thus require several arcs to open and close simultaneously. The following two sub-sections describe the neighbour solution evaluation when closing and opening an arc respectively.

4.4.1. Closing arc neighbour move

Closing arc *(i,j)* means changing the design vector for an incumbent solution, Y_{n-1}. The incumbent solution's flow vector, X_{n-1}, is no longer feasible, if arc *(i,j)* has positive flow, because the commodities traversing arc *(i,j)*, $\mathcal{P}_{ij}:\{p \in \mathcal{P} \mid x_{ij}^p > 0\}$, no longer have feasible flow paths. Figure 7 shows a network with flow of two commodities (light and dark grey). Closing arc *(i,j)* results in the dark grey commodity no longer having a feasible flow path.

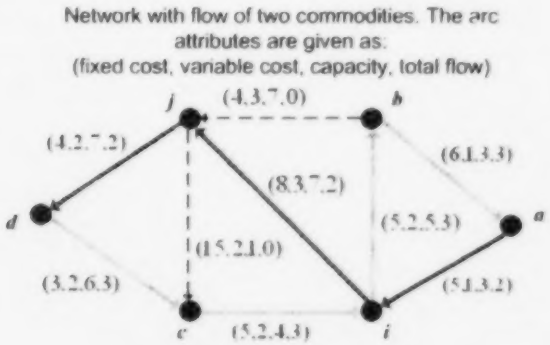

FIGURE 7

In order to maintain flow feasibility, the commodities on arc *(i,j)* are redirected onto other arcs. They may be redirected on open arcs, using those arcs' residual capacity, $r_{ij} = u_{ij} - \sum_{p \in \mathcal{P}} x_{ij}^p$, or, to e.g. enable the restoration of broken connectivity, on closed arc that in turn are opened. The redirection of the commodities may be done either as an aggregated entity or individually for each commodity. The advantage of redirecting commodities as an aggregated entity is that computational effort may be saved. However, redirecting commodities individually allows for a more efficient use of residual capacity on open arcs, as the flow entities are smaller.

The redirection is performed by constructing a Γ^p-residual graph, $\mathcal{G}_p^\Gamma = (\mathcal{N}, \mathcal{A}_p^\Gamma)$, where $\Gamma^p = w^p$ is the total volume of commodity $p \in \mathcal{P}$. All open arcs, except arc *(i,j)*, are included in the residual graph if their residual capacity, r_{kl}, is larger than Γ^p:

$$\mathcal{A}^{\Gamma,open} = \{(k,l) \in \mathcal{A} \mid y_{kl} = 1 \wedge r_{kl} = u_{kl} - \sum_{p \in \mathcal{P}_{ij}} x_{kl}^p \geq \Gamma^p\}$$

All closed arcs are included in the residual graph if their capacity u_{ij} is larger than Γ^p:

$$\mathcal{A}^{\Gamma,closed} = \{(k,l) \in \mathcal{A} \mid y_{kl} = 0 \wedge u_{kl} \geq \Gamma^p\}$$

The union of the two arc sets makes up the arc set in the residual graph, $\mathcal{A}_p^\Gamma = \mathcal{A}_p^{\Gamma,open} \cap \mathcal{A}_p^{\Gamma,closed}$. Notice that the residual capacity on an arc is calculated without flows of the commodities in $\mathcal{P}_{ij}$. This means that the flows on any arc of the commodities in $\mathcal{P}_{ij}$ are removed entirely from the network, and new path must be found from their origin nodes to their destination nodes. To limit computational complexity, only one path is determined for a commodity by solving the shortest path on its residual graph. The residual graph arc costs used to solve the shortest path are:

$$c_{kl}^\Gamma = \begin{cases} f_{kl} + c_{kl} \cdot \Gamma^p & \text{if } (k,l) \in \mathcal{A}^{\Gamma,closed} \\ c_{kl} \cdot \Gamma^p & \text{if } (k,l) \in \mathcal{A}^{\Gamma,open} \end{cases}$$

The cost of using a closed arc represents the cost of opening the arc, f_{kl}, and the cost of routing the commodity. The cost of using an open arc is alone the cost of routing the commodity. Solving the shortest path, π^p, for a commodity $p \in \mathcal{P}_{ij}$ results in a path with the set of arcs, $\mathcal{A}_p^\Gamma(\pi^p)$. The subset of arcs belonging to the arc set $\mathcal{A}_p^\Gamma(\pi^p)$ that require the original arcs to be opened are denoted ΔY_p^π:

$$\Delta Y_p^\pi = \{(i,j) \in \Delta Y_p^\pi \mid (i,j) \in \mathcal{A}_p^\Gamma(\pi^p) \wedge y_{ij} = 0\}$$

The total cost of using path π^p can be expressed as:

$$\Delta Z_p^\Gamma = \sum_{(i,j) \in \mathcal{A}_p^\Gamma(\pi^p)} c_{ij}^\Gamma$$

Figure 8 shows the residual graph for the dark grey commodity using arc *(i,j)* from figure 7. Notice e.g. that arc *(j,c)* is not included in the residual graph, as its capacity (u_{jc}=1) is less than the required redirection flow ($\Gamma^p = 2$). Similarly arc *(b,a)* is not included because its residual capacity is zero ($r_{kl} = 0$). The shortest path (and coincidentally the only feasible path) in the residual network is *(a,i)*, *(i,b)*, *(b,j)*, and *(j,d)* and requires the opening of arc *(b,j)* in the original network.

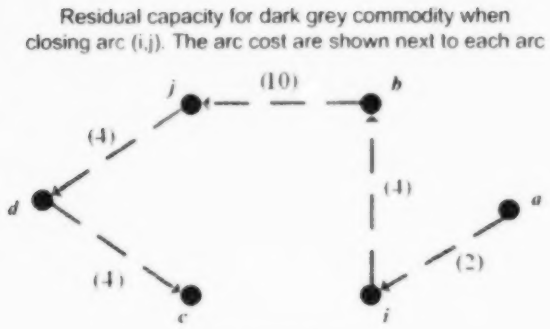

FIGURE 8

Figure 9 shows the updated network from figure 7 where arc (i,j) has been closed.

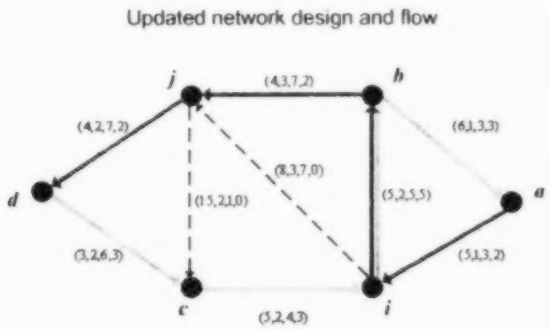

FIGURE 9

Because the design changes if a closed arc is used in the path, and because the commodities cannot use the same residual capacity on other open arcs, the flow diversions onto the original network arcs needs to be updated after redirecting each commodity $p \in \mathcal{P}_{ij}$. The redirection needs to be done sequentially for each commodity. The sequence in which commodities are redirected is here done according to the enumeration of the commodities. Whenever a commodity has been redirected, it is removed from the set $\mathcal{P}_{ij}$.

To calculate the total system cost of the potential neighbour solution after closing arc *(i,j)* the procedure summarized in figure 10 is used. The procedure starts by initializing the total system cost to be equal that of the incumbent solution minus the fixed cost, f_{ij}, of arc *(i,j)* (that is to be closed) and minus the routing cost of commodities $\mathcal{P}_{ij}$ on all arcs. For each redirection of commodities the neighbour solution's total system cost is updated by adding the new path cost. The design vector is updated with the arcs to be opened, ΔY_p^{π} and the flow vector is updated with the flow of the new commodity path.

Repeating the update of the flow and design vectors results in an approximation for the neighbour solution $(\bar{Y}_n, \bar{X}_n)$ and in an estimated potential neighbour total system cost $\bar{Z}_n$ which is used for the neighbour solution value comparison.

$$
\textit{Initialize}: \begin{cases} \widetilde{Z}_n := Z_{n-1} - f_{ij} - \sum_{p \in \mathcal{P}_{ij}(i,j) \in \mathcal{A}} \sum c_{ij}^p x_{ij}^p \\ \widetilde{Y}_n := Y_{n-1}, \text{ except arc (i, j)} \\ \widetilde{X}_n := X_{n-1}, \textit{except } x_{ij}^p = 0 \ \forall \ p \in \mathcal{P}_{ij}, (i,j) \in \mathcal{A} \end{cases}
$$

While $\mathcal{P}_{ij} \notin \varnothing$

Construct residual graph Γ^p for commodity $p \in \mathcal{P}_{ij}$

Find shortest path π^p to determine ΔZ_p^{Γ} *and* ΔY_p^{π}

$$
\text{Update}: \begin{cases} \widetilde{Z}_n = \widetilde{Z}_n + \Delta Z_p^{\Gamma} \\ \widetilde{Y}_n = \widetilde{Y}_n + \Delta Y_p^{\pi} \\ \widetilde{X}_n = \widetilde{X}_n, \textit{set } x_{ij}^p = \Gamma^p \ \forall (i,j) \in \mathcal{A}_p^{\Gamma}(\pi^p) \\ \text{remove } p \text{ from } \mathcal{P}_{ij} \end{cases}
$$

FIGURE 10

4.4.2. Opening arc neighbour move

Opening an arc can improve node imbalances of an incumbent solution and thereby move the search towards a feasible neighbour solution, as illustrated by removing arc *(i,j)* in figure 5. Therefore it is advantageous to include a move in the neighbourhood structure that adds an arc to the design vector.

The question that needs to be addressed is what approach to use when calculating the total system cost following the addition of an arc to the design vector. Opening an arc may result in new routing possibilities for commodities and thus change the flow vector and require a re-computation of the CMCF problem to determine the flow vector accurately. However, as stated previously, it may also have little or no impact on the flow vector. Therefore we assume that making the exact computation of the flow vector for each potential candidate by solving the CMCF is computationally intractable if a fast algorithm is to be achieved.

Alternatively, an approximation to the total system cost has to found. Consider the following; by opening an arc the fixed cost associated to it is incurred and must be added to the total system cost. For the new design vector, denoted Y_n, it is clear that the incumbent flow vector, X_{n-1}, is feasible. The optimal flow vector, X_n, for the design vector Y_n could possibly be different given the new routing possibilities the opened arc provides. However, the arc will only be used by the commodities if it results in a reduction in total system cost, $Z(Y_n, X_n) \leq Z(Y_n, X_{n-1})$. We thus know that the "real" total system cost for a design change $Y_{n-1} \rightarrow Y_n$ is bounded upwards by $Z(Y_n, X_n) \leq Z(Y_{n-1}, X_{n-1}) + f_{ij}$, where f_{ij} is the fixed cost of the arc *(i,j)* that is opened. Establishing a lower bound for the candidate solution total system cost is not as simple. However, recall the interpretation of an arc from the introduction as being a service between two terminals in time. The fixed cost associated to the arc is thus interpreted at the cost of offering the service (equipment use, crew cost, access right, running empty fuel cost etc.). These costs are arguably much higher than the variable cost associated to an arc (loaded unit fuel costs, unit handling cost etc.). The

potential reduction in the variable cost component of the total system cost would thus be less significant than the cost incurred by opening an arc. We thus assume that the following approximation is reasonable:

$$Z(Y_n, X_n) \approx Z(Y_{n-1}, X_{n-1}) + f_{ij}$$

Assuming the approximation is reasonable it then follows that updating the flow vector $X_{n-1} \rightarrow X_n$ does not affect the total system cost significantly. We therefore use the approximation that the candidate solutions flow vector, $\tilde{X}_n$, is equal to the incumbent solution flow vector:

$$\tilde{X}_n = X_{n-1}$$

Given the candidate solution design vector as $Y_n = Y_{n-1} \cup (i,j)$, the total system cost for the candidate solution can be calculated by adding the fixed cost of arc *(i,j)* to the total system cost of the incumbent solution:

$$\tilde{Z}_n = Z_{n-1} + f_{ij}$$

By ignoring the potential lower variable cost it yields that $\tilde{Z}_n \geq Z_n$. The clear advantage of using this simple approximation is the minimal computational requirements necessary to calculate the estimated total system cost $\tilde{Z}_n$ resulting in a very fast evaluation of each arc candidate.

4.5. Determining a candidate list for the exploration phase

The basic add/drop move has $|\mathcal{A}|$ potential neighbour solutions to examine in an exhaustive search of the neighbourhood. However, that may require an examination of uninteresting moves resulting in waste of computational time. It is more effective to consider only a limited number of "good" potential solutions. A list of candidate arcs, that is a subset of the network arcs, $\mathcal{C}_1^{n-1} \subseteq \mathcal{A}$, is composed for an incumbent solution $n-1$.

In the implementation proposed here we have opted to determine the candidate list from four candidate sub-lists each ordered according to different criteria. The four sub-lists are:

- $\mathcal{C}_f$, open arcs with the highest fixed cost
- $\mathcal{C}_v$, open arcs with the highest variable cost
- $\mathcal{C}_r^n$, open arcs with highest residual capacity
- $\mathcal{C}_p^n$, Arcs with the lowest estimated penalty value $\tilde{P}_n$

The length of each sub-list is determined by the empirical parameters l_f, l_v, l_r, and l_p. In the implementation these are given externally and remain constant throughout the progress of the algorithm.

The motivations behind the four criteria are as follows. Closing an arc with a high fixed cost can reduce the total system cost. Similarly closing an arc with high variable cost can decrease the total system cost. Arcs with high residual capacity do not use their capacity to its full extent and may therefore have a high marginal fixed cost per unit of flow. Closing an arc with large residual capacity can thus remove the fixed cost incurred by the arc while only having to redirect a small amount of flow onto other paths. Choosing an arc, that may be either open or closed, with a low expected penalty value, will decrease the solution value and thus move the search towards a feasible solution.

The order in the fixed cost sub-list e_f and the variable cost sub-list e_v does not change and can be initialized at the start of the algorithm. The residual capacity sub-list e_r^n needs to be updated after each iteration. The penalty sub-list e_p^n also needs to be updated every time a new design is adopted. The impact of closing an arc may involve opening other arcs, and because of the interdependency between opening and closing arcs on the node imbalances, it is not possible to determine the exact penalty value of a potential neighbour solution when establishing the candidate list for an incumbent solution. Instead, an estimate to determine the penalty value, and in turn the order of the sub-list is used based on the simplifying assumption that the design only changes by adding or dropping the candidate arcs, although closing an arc may involve opening other arcs. From this approximation the resulting system imbalance, $\tilde{\psi}_n^{\Re}$, is easily computed based on the incumbent solution's system imbalance, $\psi_{n-1}^{\Re}$, by updating the imbalance changes in the candidate arc's two nodes:

$$\tilde{\psi}_n^{\Re} = \psi_{n-1}^{\Re} + \Delta(\psi_n^i) + \Delta(\psi_n^j),$$
$$\text{where } \Delta(\psi_n^i) = \left|\psi_n^i - \psi_{n-1}^i\right|$$
$$\text{and } \Delta(\psi_n^j) = \left|\psi_n^j - \psi_{n-1}^j\right|$$

Finding the potential neighbour solution's estimated maximal node imbalance, $\tilde{\psi}_n^{\max}$, is more complicated. In order to have a correct value for the maximal imbalance it is not sufficient to consider the changes in the candidate arc's two nodes as is the case with the total system imbalance. Let us assume that node i has the maximal node imbalance in the incumbent solution, $\left|\psi_{n-1}^i\right| = \psi_{n-1}^{\max}$, and that arc (i, j) is closed. Let us further assume that the absolute value of the node imbalance in node i is smaller in the potential neighbour solution than in the incumbent solution, $\left|\psi_n^i\right| < \left|\psi_{n-1}^i\right|$. There is no guarantee that the maximal node imbalance is equal to the node imbalance in node i for the potential neighbour solution, as there might be another node k with the same maximal imbalance in the incumbent solution, $\left|\psi_{n-1}^k\right| = \left|\psi_n^k\right| = \left|\psi_{n-1}^i\right| = \psi_n^{\max}$. The maximal imbalance will in that case not change for the potential solution, even though the imbalance in node i become smaller. To determine the real value of the maximal node imbalance, all the nodes in the network would need to be examined for a potential neighbour solution. To limit computational requirements, $\tilde{\psi}_n^{\max}$ is estimated using the following approximation:

$$\tilde{\psi}_n^{\max} = \begin{cases} \psi_{n-1}^{\max} & \text{if } \left|\psi_{n-1}^i\right| < \psi_{n-1}^{\max} \text{ and } \left|\psi_{n-1}^j\right| < \psi_{n-1}^{\max} \\ \max(\psi_n^i, \psi_n^j) & \text{otherwise} \end{cases}$$

The approximation assumes correctly that the maximal node imbalance does not change if the node imbalances in node i and j in the incumbent solution are both smaller than the maximal node imbalance. If however, the imbalance in node i or j in the incumbent solution is equal to the maximal node imbalance, $\psi_{n-1}^{\max}$, the value may change. If the maximal node imbalance increases the new maximal node imbalance is calculated correctly by the approximation. If the node imbalance decreases there is, as explained, no guarantee that there is not another node in the network with the maximal node imbalance. However, to reward the decrease of a (large) node imbalance the estimated penalty cost is calculated as if the maximal node imbalance for the network has decreased.

The estimated penalty value of a potential neighbour solution based on the potential system imbalance and the estimated potential maximal imbalance is calculated as:

$$\tilde{P}_n = \tilde{f} \cdot \tilde{\psi}_n^{cs} \cdot \tilde{\psi}_n^{\max}$$

The estimated total solution value for a potential neighbour solution n is calculated as the sum of the estimated total system cost (calculated as shown in sections 4.4.1 and 4.4.2) and the estimated penalty value:

$$\tilde{V}_n = \tilde{Z}_n + \tilde{P}_n$$

The candidate arcs are ordered in the penalty sub-list $\mathcal{C}_p^n$ based on the approximated solution value.

4.6. Tabu list composition for the exploration phase

The purpose of the Tabu list is to avoid cycling between neighbour solutions. The Tabu list contains information on the previous l_t neighbour solutions. As the name indicates, solutions on the Tabu list may not be considered as a potential neighbour solution.

It is not practical to save all the attributes (design and flow vectors) for a solution, and thus an alternative is considered. Since the basic neighbour move is based on flipping the value of the design variable of arc (i, j) a Tabu move is defined as choosing the same arc to be flipped back within the span of the following l_t iterations. The Tabu list can thus be designed as a list holding the previous l_t chosen candidate arcs.

Keeping only information on chosen candidate arcs in the Tabu list, as opposed to the entire design and flow vectors may result in banning non-Tabu neighbour solutions. If an arc is added to the Tabu-list at iteration n it will stay there until iteration $n+l_t$ and thus may not be considered as a candidate arc. However, at iteration m, $n<m<n+l_t$, adding the chosen candidate arc from iteration n does not necessarily result in the same neighbour solution, because of the changes made to the solution between iteration n and m. In order not to strictly exclude arcs from good potential solutions they are only Tabu'ed from the candidate list. That means that if arc (i, j) is Tabu it can still re-enter into the solution if it is chosen on the redirection path of a commodity (as explained in section 4.4.2).

The Tabu list is initialized as an empty list, $\mathcal{T} = \{\emptyset\}_t$. For iterations $n \le l_t$ the chosen candidate arc in iteration n, $c_n \in \mathcal{C}^n$, is added to the Tabu list $\mathcal{T} = \mathcal{T} \cap c_n$. For iterations $n > l_t$ the first element of the Tabu list, the candidate arc of iteration $n > l_t$, c_{n-l_t}, is removed and the candidate arc of current iteration n, $c_n \in \mathcal{C}^n$ is pushed into the back of the Tabu list to get $\mathcal{T} = \mathcal{T} \setminus c_{n-l_t} \cup c_n$.

4.7. Neighbourhood structure for the feasibility phase

The neighbourhood structure of the exploration phase allows moves between infeasible solutions with respect to the design balance constraints. There is therefore no guarantee that the exploration of the solution space in phase 1 will result in feasible solutions. The feasibility phase is designed specifically to find feasible solutions.

At the termination of the exploration phase an infeasible incumbent solution c_{n-1} is available where a number of nodes are imbalanced. Consider once more the network design in figure 5. Assuming arc (i, j) is not included in the design vector, the solution is feasible with respect to the design balance constraints. If arc (i, j) is added the solution is now infeasible because there are imbalances in node i and j. More specifically the imbalance of node i is -1 and the imbalance of node j is +1. It is easily seen that for any imbalanced solution, the sum of the absolute negative imbalances is always equal to the sum of the

positive imbalances. The addition of an arc (i, j) to the design vector will decrease the imbalance by one in origin node i and increase the imbalance by one in destination node j.

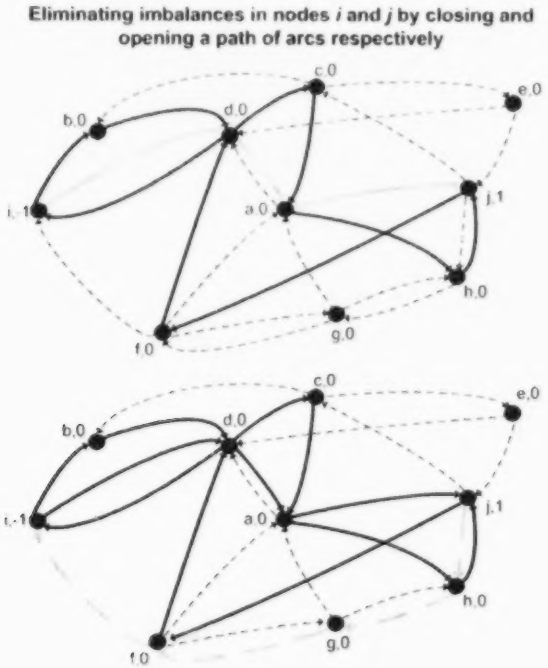

FIGURE 11A & B

The idea behind the neighbourhood structure for the feasibility phase is to determine a pair of oppositely imbalanced nodes, to close or open a path of arcs between them, and thereby reducing the imbalances in the two nodes. Closing a path of arcs from a node with negative imbalance to a node with a positive imbalance will reduce the imbalance in each of the nodes. Similarly, opening a path of arcs from a node with a positive imbalance to a node with negative imbalance will also reduce the imbalance in both nodes. Figure 11a & b show examples of connecting two nodes i and j by either closing or opening a path of arcs respectively. The imbalances of nodes lying on the path in between are unaffected because both an ingoing and an outgoing arc will either be removed or added to the node. Feasible solutions with respect to design can be achieved by closing the arcs on path (i,d), (d,a) and (a,j) as shown with the grey arcs in figure 11a, or by opening the arcs on path (j,h), (h,g), (g,f) and (f,i) as shown in figure 11b. Iteratively matching oppositely imbalanced nodes and closing or opening paths of arcs between them eventually reduces the imbalances to zero in all the nodes in the network and generates a feasible solution.

4.8. Neighbourhood move and neighbour solution evaluation for the feasibility phase

The template for the neighbourhood moves for the feasibility phase is the add/drop procedure. Instead of adding or dropping a single arc, as in the exploration phase, the add/drop procedure is performed on paths of arcs. To achieve a feasible solution we iteratively match pairs of oppositely signed imbalanced nodes and connect with paths to close or open. If the procedure is successful $\psi^{\mathfrak{R}}/2$ iterations are performed,

where $\psi^{\mathfrak{N}}$ is the total network imbalance, equivalent reducing the imbalance in two nodes within each iteration (one negative and one positive).

To match the nodes the following procedure is adopted.

- First, The maximum imbalance node is identified, $i:\{|\psi^i| = \psi^{\max}\}$
- Second, the set of nodes with oppositely signed imbalances are identified, $\mathfrak{N}^c:\{j \in \mathfrak{N}^c | \psi^i \cdot \psi^j < 0\}$.
- Third, a candidate set, $\mathcal{C}_2^{n-1}$, of path (to open or close) is then identified between node i and any of the nodes in $\mathfrak{N}^c$. The candidates are evaluated at the best path (to open or close) is selected as the neighbourhood move.

Opening or closing a path of arcs in the design may have a big impact on the flow solution. If a path is closed, many commodities may have to be redirected using the residual capacity of other open arcs. Opening a path may result in better routing possibilities for commodities. To get a reasonable evaluation of the impact on the flow, the CMCF problem is solved for each candidate path in $\mathcal{C}_2^{n-1}$. The design and flow vectors obtained by adding or removing the path resulting in the smallest total system cost, Z_c^*, is implemented resulting in a new incumbent solution.

4.9. Determining a candidate list for the feasibility phase

For a single pair of imbalanced nodes there can be several possible paths that can be opened or closed between them. E.g. in figure 11a & b there are several paths that could be opened or closed and result in a design feasible solution. If there are several imbalanced nodes the number of possible paths between positive and negative imbalanced nodes is large. Furthermore the evaluation of each potential path is done by solving the CMCF problem, which for large instances is a costly procedure with respect to computational time. We thus restrict the number of examined candidate paths to a few interesting ones. We propose a candidate list containing four candidate paths. Each of the four candidate paths are found by solving the shortest path problem between node $i:\{|\psi^i| = \psi^{\max}\}$ and nodes $j \in \mathfrak{N}^c$, or vice versa, on four modified networks with different customized arc costs. For each of the four modified networks the shortest of the calculated shortest paths are selected as candidate paths. The four modified networks and arc costs are:

- The network, $\mathcal{G}^1 = (\mathfrak{N}, \mathfrak{A}^1)$, is built of the set of open arcs:
- $\mathfrak{A}^1 : \{(i,j) \in \mathfrak{A}^1 \,|\, (i,j) \in \mathfrak{A} \wedge y_{ij} = 1\}$.
- The arc costs are set to the arcs' own total flow values:
- $c_{ij}^1 = \sum_{p \in \mathcal{P}} x_{ij}^p \;\; \forall (i,j) \in \mathfrak{A}^2$
- The candidate path on this network, π^1, will, due to the nature of the arc costs, have little flow on it. Closing this path will require only a relatively small amount of commodity flow to be redirected. Closing this path leaves a significant chance of finding a feasible flow vector whilst saving the fixed costs of the arcs.
- The network, $\mathcal{G}^2 = (\mathfrak{N}, \mathfrak{A}^2)$, is built of the set of open arcs:
- $\mathfrak{A}^2 : \{(i,j) \in \mathfrak{A}^2 \,|\, (i,j) \in \mathfrak{A} \wedge y_{ij} = 1\}$.

- The arc costs are set equal to the largest fixed cost of all arcs in $\mathcal{A}$ minus the fixed cost of arc (i,j):
- $c_{ij}^2 = \max_{(k,l)\in\mathcal{A}} (f_{kl}) - f_{ij} \;\forall\, (i,j)\in \mathcal{A}^2$
- The candidate path from this network, π^2, will consist of high cost arcs. Closing the arcs on this path will remove the high fixed costs which will reduce the total system cost significantly.
- The network, $\mathcal{S}^3 = (\mathcal{N}, \mathcal{A}^3)$, is built of the set of inversed closed arcs:
- $\mathcal{A}^3 : \{(i,j)\in \mathcal{A}^3 \mid (j,i)\in \mathcal{A} \wedge y_{ij} = 0\}$.
- The arc costs are set to equal the variable cost of the arcs in $\mathcal{A}$:
- $c_{ij}^3 = c_{ji} \;\forall\, (i,j)\in \mathcal{A}^3$
- The candidate path from this network, π^3, will consist of arcs with small variable costs. Opening this path may enable some cheaper routing alternatives for commodities and thereby reducing the variable cost component of the total system cost although fixed costs are incurred by opening the arcs in the path.
- The network, $\mathcal{S}^4 = (\mathcal{N}, \mathcal{A}^4)$, is built of the set of inversed closed arcs:
- $\mathcal{A}^4 : \{(i,j)\in \mathcal{A}^4 \mid (j,i)\in \mathcal{A} \wedge y_{ij} = 0\}$
- The arc cost are set to equal the fixed cost of the arcs in $\mathcal{A}$:
- $c_{ij}^4 = f_{ji} \;\forall\, (i,j)\in \mathcal{A}^4, (j,i)\in \mathcal{A}$
- The candidate path on this network, π^4, will consist of arcs with low fixed cost. Opening this path will result in an increase in the total system cost and is included as a "last resort" candidate to reduce the absolute imbalances by opening cheapest possible path.

Given that the candidate list only examines a subset of four paths, there is no guarantee that it is possible to determine a new incumbent solution. If no feasible solution is found from any of the candidates in $\mathcal{C}_2^{n-1}$, the feasibility phase is terminated and the exploration phase continues from the infeasible incumbent solution obtained thus far in the feasibility phase.

There could be other interesting candidate paths, possibly found on other constructed graphs with other cost structures or alternative path finding methods. However, as shall be shown in the results section, limiting the candidate list to the 4 proposed paths is more than adequate in the algorithm. Proposition and experimentation with alternative candidate list structures is left to future research.

Size and characteristics of selected network instances for paramter tuning

instance name	network size			Network characteristic		
	nodes	arcs	commodities	capacity ratio	cost ratio	Opt. Solution
R10,F05,C2	20	120	40	medium	medium	yes
R12,F10,C2	20	120	200	medium	high	yes
R13,F01,C8	20	220	40	tight	low	yes
R15,F10,C8	20	220	200	tight	high	yes
C20,230,200,F,L	20	230	200	loose	high	no
R16,F10,C1	20	314	40	loose	high	no
R17,F01,C1	20	318	100	loose	low	yes
R18,F05,C2	20	315	200	loose	medium	no
C30,520,100,V,T	30	519	100	tight	low	no
C100,400,30,F,L,10	100	400	30	loose	high	no

Table 1

5. Computational results

The performance of the algorithm is tested on the network design instances used in [Crainic et al 2000] and [Ghamlouche et al 2003]. There are two sets of instances, R-problems and C-problems, out of which a number of the most difficult ones have been selected for the final experimentation. Both sets of instances are general transhipment networks with no parallel arcs and one commodity per origin-destination pair. The instances differ in size (nodes, arcs, commodities). Furthermore the fixed cost, variable cost, and capacity vary in relative importance for each of the instances. The same unit cost on arcs is used for all commodities.

In each of the network design instances the arcs have varying capacities. It must be presumed that in a scheduled service network design application, the capacities are somewhat homogenous. However, intuitively it can be assumed that it is more difficult to solve the DBCMND model on networks with heterogeneous capacities because of the fixed cost/capacity trade-off. We thus assume the results presented here are not an over-estimation of the performance of an eventual application of the algorithm on for a scheduled service design network.

For the computations in the TS algorithm the shortest path instances is solved with the ML-Thresh-X2 form Jørgensen et al. (2004) while the CMCF problems are solved using the Xpress-MP LP-solver. The algorithm is implemented in C++ using Microsoft Visual Studio .NET 2003 using the Xpress-BCL builder component library. The TS-algorithm's performance is compared to solutions obtained with the Xpress-MP MIP-solver by running both with a 3600 seconds time limit. All computations are performed on a laptop computer with a 2.26 GHz Intel Pentium 4 processor.

5.1. Initial parameter tuning

There are eight parameters in the TS algorithm that need to be determined (see the list in table 2). In order to find an adequate parameter configuration, an initial parameter tuning experiment was conducted. The goal of the parameter tuning experiment was to find a single parameter configuration that could be applied on all instances regardless of their network characteristics. In order to find a robust parameter configuration, ten instances were selected from both the C and R problem sets to represent the various possible network characteristics on which the parameter tuning experiment was conducted. These instances and their characteristics are listed in table 1. The 'capacity ratio' column is a ratio of commodity demand over total network capacity, where a 'tight' capacity ratio signifies limited capacity in relation to the total demand and 'loose' capacity a surplus of capacity. The 'cost ratio' column is a ratio of the fixed cost over the variable cost of arcs. The 'Opt. solution' column indicates whether an optimal solution was found for the instance solving the problem within a 3600s time limit using the Xpress-MP MIP solver. As can be seen the instances represent various combinations of characteristics and is thus assumed to be representative of all the instances in the data sets.

The parameter tuning experiment was conducted using a 2-level 2^8 factorial experimentation plan (Montgomery (2000)) where each of the eight parameters were evaluated on two levels. The details of the experiment can be found in appendix A. The results of the experiment gave indications of a robust algorithm with respect to the parameter settings; a claim that will be further supported by the computational results in the following section.

Parameter	Description	Value
I_f	# of Fixed cost candidates	5
I_p	# of Penalty cost candidates	5
I_r	# of Residual capacity candidates	15
I_v	# of Variable cost candidates	15
I_t	Tabu list length	25
ψ^p	Penalty value scale	0.5
I_g	Improvement gap	5%
I_i	Improvement range	10

TABLE 2

Computational results (3600s.) on C-problems									
Instance	Nodes	Arcs	Com.	XpressMP	Bound	XP/Bound GAP	TS	TS/XP GAP	TS/Bound GAP
C20,230,200,V,L	20	228	200	101.112	86.180	14,77%	102.919	1,76%	16,26%
C20,230,200,F,L	20	230	200	153.534	122.311	20,34%	150.764	-1,84%	18,87%
C20,230,200,V,T	20	229	200	105.840	92.608	12,50%	103.371	-2,39%	10,41%
C20,230,200,F,T	20	228	200	154.026	124.358	19,26%	149.942	-2,72%	17,06%
C20,300,200,V,L	20	294	200	81.184	73.894	8,98%	82.533	1,63%	10,47%
C20,300,200,F,L	20	292	200	131.876	110.533	16,18%	128.757	-2,42%	14,15%
C20,300,200,V,T	20	291	200	78.675	74.583	5,20%	78.571	-0,13%	5,08%
C20,300,200,F,T	20	291	200	127.412	106.628	16,31%	116.338	-9,52%	8,35%
C30,520,100,V,L	30	518	100	55.138	54.160	1,77%	55.981	1,51%	3,25%
C30,520,100,F,L	30	516	100	n/a	92.636	n/a	104.533	n/a	11,38%
C30,520,100,V,T	30	519	100	53.125	52.681	0,84%	54.493	2,51%	3,33%
C30,520,100,F,T	30	517	100	106.761	97.653	8,53%	105.167	-1,52%	7,14%
C30,520,400,V,L	30	520	400	n/a	111.054	n/a	119.735	n/a	7,25%
C30,520,400,F,L	30	520	400	n/a	143.335	n/a	162.360	n/a	11,72%
C30,520,400,V,T	30	516	400	n/a	114.725	n/a	120.421	n/a	4,73%
C30,520,400,F,T	30	518	400	n/a	148.210	n/a	161.978	n/a	8,50%
C30,700,100,V,L	30	680	100	48.849	48.400	0,92%	49.429	1,17%	2,08%
C30,700,100,F,L	30	680	100	65.516	59.483	9,21%	63.889	-2,55%	6,90%
C30,700,100,V,T	30	687	100	47.052	46.260	1,68%	48.202	2,39%	4,03%
C30,700,100,F,T	30	686	100	57.447	55.123	4,05%	58.204	1,30%	5,29%
C30,700,400,V,L	30	685	400	n/a	94.725	n/a	103.932	n/a	8,86%
C30,700,400,F,L	30	679	400	n/a	128.950	n/a	157.043	n/a	17,89%
C30,700,400,V,T	30	678	400	n/a	95.183	n/a	103.085	n/a	7,67%
C30,700,400,F,T	30	683	400	n/a	128.441	n/a	141.917	n/a	9,50%

TABLE 3

5.2. TS Algorithm vs. MIP-solver performance

From the results of initial parameter tuning we determined a single parameter configuration to be used for testing the algorithm performance. The chosen parameter values are shown in table 2. The TS-algorithm is tested on instances from the R and C data sets. The 24 most difficult instances have been selected from the C data set and the 54 most difficult ones from the R data set. The TS algorithm ad the Xpress-MP MIP solver have been applied to each of the 78 sets with a 3600 seconds CPU-time limit using a PC with an Intel Pentium 4, 2.26Ghz processor.

The computational results for the selected C instances are shown in table 3 and the selected R instances in table 4. The two tables are composed identically as such. The 'XpressMP' column shows the results obtained with the Xpress-MP MIP solver within the 3600 seconds computational time limit. The 'Bound' column shows the lower bound of the instance obtained by the MIP-solver and the 'XP/Bound GAP' column gives the relative gap between the MIP solution and its lower bound. The 'TS' column shows the results obtained using the TS algorithm using the parameter setting showed in table 2 within the 3600 seconds computational time limit. Finally the 'TS/XP GAP' and the 'TS/Bound GAP' columns show the relative gaps between the TS algorithm results and the MIP-solver solution and TS algorithm results and the lower bound respectively. An 'n/a' signifies that no integer solution was obtained within the time limit. Note that for the 9 of the C problems and the 2 of the R problems, where the MIP-solver failed to find a feasible integer solution, the TS algorithm succeeded. The TS algorithm ability to find feasible solutions for all 78 instances gives a first indication of robustness.

Computational results (3600s.) on R-problems									
Instance	Nodes	Arcs	Com.	XpressMP	Bound	XP/Bound GAP	TS	TS/XP GAP	TS/Bound GAP
R13,F01,C1	20	220	40	147349	147349	0,00%	147837	0,33%	0,33%
R13,F05,C1	20	220	40	281283	269793	4,08%	281668	0,14%	4,22%
R13,F10,C1	20	220	40	404045	365588	9,52%	404434	0,10%	9,61%
R13,F01,C2	20	220	40	155887	155887	0,00%	159852	2,48%	2,48%
R13,F05,C2	20	220	40	295180	292249	0,99%	311209	5,15%	6,09%
R13,F10,C2	20	220	40	443831	411984	7,18%	470034	5,57%	12,35%
R13,F01,C8	20	220	40	218787	218787	0,00%	225339	2,91%	2,91%
R13,F05,C8	20	220	40	502811	480366	4,46%	512027	1,80%	6,18%
R13,F10,C8	20	220	40	812606	764751	5,89%	875984	7,24%	12,70%
R14,F01,C1	20	220	100	422709	422709	0,00%	431562	2,05%	2,05%
R14,F05,C1	20	220	100	835597	717432	14,14%	811102	3,02%	11,55%
R14,F10,C1	20	220	100	1259890	954048	24,28%	1193950	5,52%	20,09%
R14,F01,C2	20	220	100	452591	452591	0,00%	465762	2,83%	2,83%
R14,F05,C2	20	220	100	912189	848873	6,94%	942678	3,23%	9,95%
R14,F10,C2	20	220	100	1397100	1210230	13,38%	1401880	0,34%	13,67%
R14,F01,C8	20	220	100	704719	699947	0,68%	720882	2,24%	2,90%
R14,F05,C8	20	220	100	1696780	1659720	2,18%	1795650	5,51%	7,57%
R14,F10,C8	20	220	100	2874660	2681490	6,72%	2997290	4,09%	10,54%
R15,F01,C1	20	220	200	1042790	969165	7,06%	1039440	-0,32%	6,76%
R15,F05,C1	20	220	200	2297560	1718650	25,20%	2170310	-5,86%	20,81%
R15,F10,C1	20	220	200	3304180	2436030	26,27%	3194270	-3,44%	23,74%
R15,F01,C2	20	220	200	1176860	1164850	1,02%	1205790	2,40%	3,40%
R15,F05,C2	20	220	200	2723740	2368130	13,06%	2698680	-0,93%	12,25%
R15,F10,C2	20	220	200	4349910	3467490	20,29%	4447950	2,20%	22,04%
R15,F01,C8	20	220	200	2402800	2392650	0,42%	2472860	2,83%	3,24%
R15,F05,C8	20	220	200	5807050	5774260	0,56%	6067350	4,29%	4,83%
R15,F10,C8	20	220	200	9169890	9025170	1,58%	10263600	10,66%	12,07%
R16,F01,C1	20	314	40	140082	140082	0,00%	142692	1,83%	1,83%
R16,F05,C1	20	314	40	259840	236810	8,86%	261775	0,74%	9,54%
R16,F10,C1	20	314	40	364786	310584	14,86%	374819	2,68%	17,14%
R16,F01,C2	20	314	40	142381	142381	0,00%	145266	1,99%	1,99%
R16,F05,C2	20	314	40	275626	251711	8,68%	277307	0,61%	9,23%
R16,F10,C2	20	314	40	396966	341107	14,07%	391386	-1,43%	12,85%
R16,F01,C8	20	314	40	180199	178497	0,94%	187176	3,73%	4,64%
R16,F05,C8	20	314	40	396721	376351	5,13%	423320	6,28%	11,10%
R16,F10,C8	20	314	40	637944	571578	10,40%	649121	1,72%	11,95%
R17,F01,C1	20	318	100	364784	362602	0,60%	374016	2,47%	3,05%
R17,F05,C1	20	318	100	730195	583094	20,15%	718135	-1,68%	18,80%
R17,F10,C1	20	318	100	1150630	769660	33,11%	1041450	-10,48%	26,10%
R17,F01,C2	20	318	100	382593	382315	0,07%	393608	2,80%	2,87%
R17,F05,C2	20	318	100	761041	695446	8,62%	786198	3,20%	11,54%
R17,F10,C2	20	318	100	1195710	971426	18,76%	1162290	-2,88%	16,42%
R17,F01,C8	20	318	100	531791	523185	1,62%	539817	1,49%	3,08%
R17,F05,C8	20	318	100	1284720	1195670	6,93%	1348750	4,75%	11,35%
R17,F10,C8	20	318	100	2047390	1927200	5,87%	2227780	8,10%	13,49%
R18,F01,C1	20	315	200	869263	772151	11,17%	864425	-0,56%	10,67%
R18,F05,C1	20	315	200	1869230	1302720	30,31%	1640200	-13,96%	20,58%
R18,F10,C1	20	315	200	2390740	1793780	24,97%	2389230	-0,35%	25,24%
R18,F01,C2	20	315	200	980178	900627	8,12%	962402	-1,85%	6,42%
R18,F05,C2	20	315	200	2146670	1730640	19,38%	1958160	-9,63%	11,62%
R18,F10,C2	20	315	200	n/a	2478890	n/a	2986000	n/a	16,98%
R18,F01,C8	20	315	200	1560790	1507840	3,39%	1617320	3,50%	6,77%
R18,F05,C8	20	315	200	4230970	3817170	9,78%	4268580	0,86%	10,58%
R18,F10,C8	20	315	200	n/a	6180620	n/a	7440780	n/a	16,94%

TABLE 4

Distribution of relative gaps between TS and XpressMP							
Data set	n/a	<[-5%]	[-5%]-[-2.5%]	[-2.5%]-[0%]	[0%]-[2.5%]	[2.5%]-[5%]	>[5%]
C	9	1	6	1	6	1	0
R	2	5	5	4	19	12	7
C+R	11	6	11	5	25	13	7

TABLE 5

To analyze the results presented in tables 3 and 4 further, we categorize the distribution of the relative gaps between the results obtained with the TS algorithm and with XpressMP's MIP-solver. Table 5 shows that for 17 out of 24 C problems and 16 out of 54 R problems the TS algorithm outperforms the MIP-solver. In aggregated figures the TS algorithm outperforms the MIP-solver in 33 out of 78 instances. These figures by themselves may not seem very convincing. However, it is important to note that the TS algorithm outperformed the MIP-solver for the large and hence more difficult instances. This statement is supported in figure 12 that shows a scatter-plot of the 78 instances with the relative gap between the MIP-solvers solutions and the lower bounds along the X-axis and the relative gap between the TS algorithm solution and the MIP-solver solution along the Y-axis. The linear regression in the figure clearly shows that the larger the relative gap between the MIP-solver solutions and the lower bounds are, the better the algorithm performs relatively. Therefore, we conclude that the algorithm performs better than the MIP-solver on large instances.

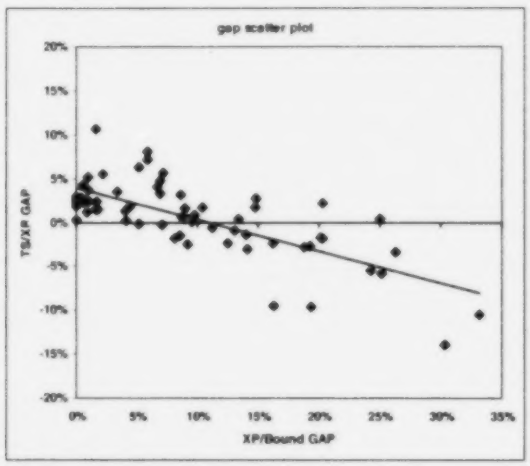

FIGURE 12

5.3. Analysis of algorithmic behaviour

We will in this section present some analysis of the TS algorithm's behaviour, in order to better determine its characteristics and applicability.

The first analysis is the performance of the algorithm in relation to the network characteristics. Figure 13 shows a graph of the relative gaps between the best feasible MIP-solver solution and the lower bound solver for the R-instances. The network characteristics are labelled along the x-axis, by increasing fixed/variable cost ratio (F01, F05, F10) and increasing "tightness" of capacity (C1, C2, C8). Each curve represents one of the 6 R-instance groupings (R13-R18) representing the network instance size. For the instances whit no feasible solution from the MIP solver, gaps of 35% have been assimilated. As for general network design problems the tendency is that the higher the fixed/variable cost ratio is the larger the gap is and thus the more difficult the instances become (the F10,CX labelled instances being the most difficult independent of the problem size). Surprisingly, the gaps peak for the loose capacity instances (labelled FXX,C1) and is lower, the tighter capacity becomes. This indicates that tight capacity DBCMND problems are easier to solve than loose ones. This tendency is presumably related to the design balance constraints. The tight capacity requires more arcs to be open in order to allow the flow of commodity

which in turn forces a high "open arcs" vs. "closed arcs" ratio to satisfy the design balance constraints. Since many arcs need to be open in the final solution the LP-relaxation is tighter than for loose capacity problems and hence results in better lower bounds.

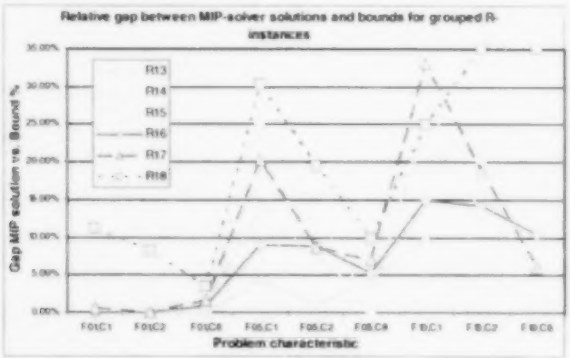

FIGURE 13

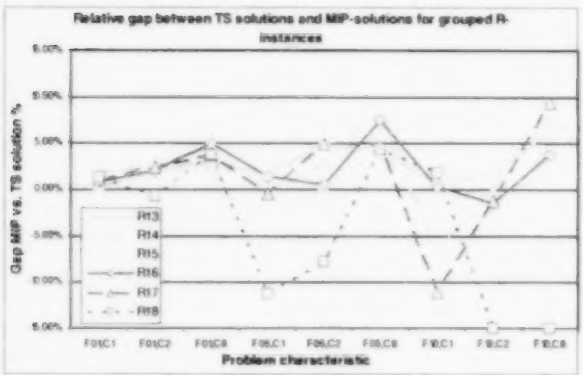

FIGURE 14

Figure 14 shows the gap between the TS algorithm solutions and the MIP-solver solutions for the R instances. Compared to the analysis of figure 13 of the poor performance of the MIP-solver on high fixed cost and loose capacity instances it is in figure 14 seen that the TS algorithm largely outperforms the MIP-solver for those instances (F05,C1 and F10,C1) for the two largest instance groups (R17 and R18). Furthermore, the TS algorithm also outperforms the MIP solver on the largest high fixed/variable cost ratio instances in R18. This indicates that the algorithm is robust with respect to instance attributes and is in particular useful for solving network instances where MIP-solvers may struggle.

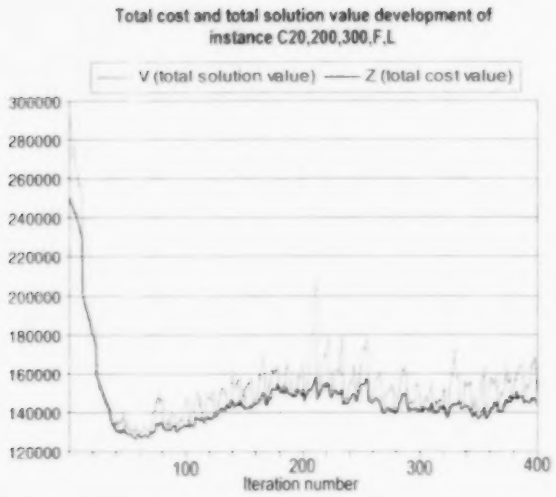

FIGURE 15

In applying the DBCMND model to service scheduling problems there are intuitively a large number of possible services to select out of which only a few are selected. Furthermore the majority of costs of operation are in the network representation captured in the fixed cost of arcs (the cost of offering the service). The variable cost, reflecting the cost from routing an amount of commodities (passengers or freight), is negligible. Thus networks for transportation service network design can be assumed to have the high fixed cost, loose capacity characteristics (considering the many possible service offerings). It therefore seems that the TS-algorithm would be the best alternative to solve the DBCMND problem applied to transport service network design problems.

In the second analysis of this section we present the behaviour of the candidate lists in both local search phases of the TS-algorithm. It is interesting to investigate if candidates are chosen from all four sub-list in the exploration phase and if all four candidates paths are chosen in the feasibility phase when running the algorithm. The results may be used to further tune the algorithm an remove eventual redundant computations. Table 6 shows the distribution of the best neighbour selection from the candidate sub-list in the explorations phase (phase 1) and the distribution of selection of four candidate paths in the feasibility phase (phase 2). The figures are averages taken over all 78 instances. As can be seen from the table, all four candidate sub-lists and path candidates contribute to the selection of the best neighbour solution. It is important to note that the actual selection distribution varies greatly for the individual instances. Especially the penalty sub-list has a tendency of not providing any best neighbour solutions for some instances. Similarly, the path candidate selection under the feasibility phase varies depending on the instance. However, since the initial objective was to provide a general tuning of the algorithm to solve DBCMND instances it is reasonable to accept these figures and conclude that all four candidate sub-lists and 4 candidate paths contribute to the algorithm. The investigation and fine-tuning of the algorithm to meet special measures and attributes for network instances is left to further research.

Candidate selection Phase 1			
l_1	l_2	l_3	l_4
33,20%	21,70%	31,80%	13,29%
Candidate selection Phase 2			
Path 1	**Path 2**	**Path 3**	**Path 4**
58,03%	12,16%	15,26%	14,55%

TABLE 6

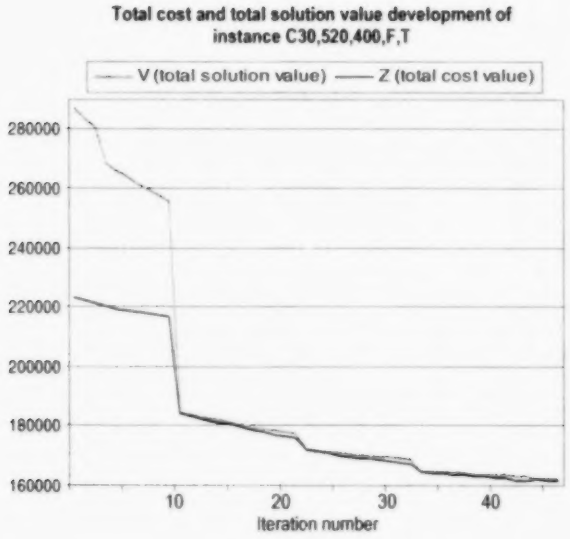

FIGURE 16

The third and last analysis we present in this section, is based on the interaction between the exploration phase and the feasibility phase and their influence on the solution value development. Figure 15 shows the development of the solution value for instance C20,200,300,F,L. The two curves represent the development in the total solution cost and the solution value (total solution cost + penalty value) by the number of iterations. As can be seen, the initial solution r-DBCMND and rounding heuristic yields a olution with high cost and penalty values. The algorithm starts off with the exploration phase and switches to the feasibility phase after eleven iterations because of the improvement gap of 5% and the iteration range value of 10. Every ten iterations the feasibility phase sets in, which can be seen as the points where the two curves meet (penalty value=0). An interesting development is that the best solution is found after 55 iterations of the exploration phase 5 initiations of the feasibility phase. For the remaining time the search oscillates above the best found solution. This could signify that the algorithm for this particular instance produces good feasible solution very fast. For the most complex instances the search is stopped before the improvement in feasible solutions smoothes out. Figure 16 shows the development of the solution value for instance C30,520,400,F,T. The drops in the curves show where the feasibility phase was run. The solution value improves for every feasibility phase run, but the search is stopped when at the time limit of 3600 s. indicating that further improvements may be achieved. The few iterations performed for this instance are a result of the several CMCF problems solved in the feasibility phase, that are very computationally demanding. For example between 500 and 1000 seconds where used for the C30,520,400,F,T instance per run of the feasibility phase depending on the level of imbalance. This raises the question of whether an alternative evaluation method should be applied in the feasibility phase to speed of the evaluations or if fewer candidate paths should be investigated. Nevertheless, although the feasibility phase is computationally demanding it also proves to be the driving force behind improving the feasible solution values and achieving good feasible solutions.

5.4. Simple parallel implementation investigation

The outcome of the initial parameter tuning was a parameter configuration that would produce the overall best results for the TS algorithm. Nevertheless, the different network characteristics may require different

parameter configurations to obtain the best possible solutions with the algorithm. We thus here examine the possibility of applying various parameter configurations to each of the instances and select the best result obtained among the configurations to see if a significant benefit could be obtained. The motivation behind this stems from the fact that the algorithm could be deployed in a parallel implementation where an instance could be solved using different parameter configurations on different CPU's simultaneously.

We have chosen to perform runs with an additional five parameter configurations. The six configurations including the one used in the previous section are shown in table 7. As can be seen only six of the eight parameters vary in the six configurations. We have chosen to fix the improvement range and improvement gap. The reason for this is due to the significant importance of the feasibility phase in finding good feasible solutions. To allow as many feasibility phase executions as possible the improvement range and the improvement gap are kept at their low and high level respectively. The remaining 6 parameters are used at various levels.

Parameter	Description	Values					
I_f	# of Fixed cost candidates	5	5	10	5	10	5
I_p	# of Penalty cost candidates	5	5	5	5	10	5
I_r	# of Residual capacity candidates	15	5	5	10	10	15
I_v	# of Variable cost candidates	15	5	5	10	10	15
I_t	Tabu list length	25	25	20	20	10	10
ω^*	Penalty value scale	0.5	0.5	2	2	2	0.5
I_g	Improvement gap	5%	5	5	5	5	5
I_i	Improvement range	10	10	10	10	10	10
	Average score	**3,14**	**3,60**	**4,01**	**4,06**	**3,71**	**2,41**

TABLE 7

The computational results from the parallel implementation can be seen in tables 8 and 9. The three columns labelled 'XpressMP', 'TS', and 'Parallel TS' show the computational results from the MIP-solver, the TS solutions with the single parameter configuration from the previous section, and the best TS solution obtained from either of the six parameter configurations respectively. The column 'Parallel improv.' shows the relative improvement between the single parameter configuration TS solution and the solution value obtained from the multiple parameter configurations. The last two columns, 'TS avg.' and 'St. dev.' show the average solution values obtained from the six parameter configurations and the standard deviation relative to the average solution values. The computational results for the individual parameter settings for the individual runs can be found in Pedersen (2006). As can be seen, some improvement can be obtained from implementing a parallel algorithm with various parameter configurations. Most improvements are small, which can be seen by 59 out 78 instances having improvements less than 2%. However, there are still instance where significant improvements are achieved. E.g. the improvement is 10.31% for instance R15,F10,C8.

Given the multiple parameter configuration results it is possible to test the initial parameter configuration in relation to the additional five configurations. The 'Average scores' row at the bottom of table 10 show how well each of the parameter configurations performed in relation to each other. For each instance a score from 1 to 6 (1 being the best) was given to each configuration based on the best obtained solution value. All configuration being equal, an average a score of 3.5 should be given to each. None of the configurations stand out as being particularly bad with configuration 4 having the worst score of 4.06. However, parameter configuration 6 with a score of 2.41 proves to be somewhat better than the first five. The only difference between the initial parameter configuration and the 6th one is the length of the tabu

list, thus indicating that the result from the initial parameter configuration pointing to a long tabu list being the most reasonable was wrong. Nevertheless, the scores also indicate that the algorithm is robust with respect to the parameter selection. This claim is further supported by investigating the standard deviations in tables 8 and 9 on the best obtained solutions with each of the six parameter configurations. Only 10 of the 74 instances showed standard deviations larger than 2%, the highest being 7,26% for instance C30,700,400,F,L. Considering the very different characteristics of the 78 instances the algorithm's robustness claim is thus supported.

Parallel computational results on C-problems						
Instance	XpressMP	TS	Parallel TS	Parallel improv.	TS avg.	St. dev.
C20,230,200,V,L	101.112	102.919	101.345	-1,55%	103.078	1,12%
C20,230,200,F,L	153.534	150.764	148.384	-1,60%	151.166	1,22%
C20,230,200,V,T	105.840	103.371	103.371	0,00%	105.559	1,85%
C20,230,200,F,T	154.026	149.942	144.786	-3,58%	147.652	1,39%
C20,300,200,V,L	81.184	82.533	80.269	-2,82%	80.951	1,07%
C20,300,200,F,L	131.876	128.757	126.258	-1,98%	128.297	1,20%
C20,300,200,V,T	78.675	78.571	78.444	-0,16%	79.749	1,[illegible]%
C20,300,200,F,T	127.412	116.338	116.338	0,00%	118.591	1,21%
C30,520,100,V,L	55.138	55.981	55.786	-0,35%	56.256	0,57%
C30,520,100,F,L	n/a	104.533	101.612	-2,87%	103.967	1,15%
C30,520,100,V,T	53.125	54.493	54.092	-0,74%	54.298	0,27%
C30,520,100,F,T	106.761	105.167	104.702	-0,44%	106.661	1,41%
C30,520,400,V,L	n/a	119.735	118.071	-1,41%	119.700	0,97%
C30,520,400,F,L	n/a	162.360	160.979	-0,86%	163.332	1,05%
C30,520,400,V,T	n/a	120.421	120.421	0,00%	121.310	0,68%
C30,520,400,F,T	n/a	161.978	161.978	0,00%	164.852	1,07%
C30,700,100,V,L	48.849	49.429	49.429	0,00%	49.723	0,45%
C30,700,100,F,L	65.516	63.889	63.292	-0,94%	63.635	0,50%
C30,700,100,V,T	47.052	48.202	47.487	-1,51%	47.969	0,52%
C30,700,100,F,T	57.447	58.204	57.187	-1,78%	58.343	1,20%
C30,700,400,V,L	n/a	103.932	103.932	0,00%	105.835	1,76%
C30,700,400,F,L	n/a	157.043	148.114	-6,03%	161.398	7,26%
C30,700,400,V,T	n/a	103.085	103.085	0,00%	103.797	0,69%
C30,700,400,F,T	n/a	141.917	138.609	-2,39%	141.743	1,17%

TABLE 8

Parallel computational results on R-problems						
Instance	XpressMP	TS	Parallel TS	Parallel improv.	TS avg.	St. dev.
R13,F01,C1	147349	147837	147.349	-0,33%	147.768	0,14%
R13,F05,C1	281283	281668	281.668	0,00%	284.719	0,92%
R13,F10,C1	404045	404434	400.656	-0,94%	405.197	0,70%
R13,F01,C2	155887	159852	156.585	-2,09%	158.638	0,87%
R13,F05,C2	295180	311209	307.180	-1,31%	310.148	0,53%
R13,F10,C2	443831	470034	437.396	-7,46%	454.649	2,83%
R13,F01,C8	218787	225339	223.541	-0,80%	225.043	0,52%
R13,F05,C8	502811	512027	510.887	-0,22%	520.720	1,49%
R13,F10,C8	812606	875984	839.174	-4,39%	870.713	2,43%
R14,F01,C1	422709	431562	427.872	-0,86%	430.679	0,47%
R14,F05,C1	835597	811102	811.102	0,00%	829.272	1,38%
R14,F10,C1	1259890	1193950	1.157.500	-3,15%	1.179.155	1,79%
R14,F01,C2	452591	485762	458.240	-1,64%	463.558	0,76%
R14,F05,C2	912189	942678	917.832	-2,71%	934.399	1,45%
R14,F10,C2	1397100	1401880	1.356.910	-3,31%	1.384.428	1,28%
R14,F01,C8	704719	720882	720.494	-0,05%	726.171	0,74%
R14,F05,C8	1696780	1795650	1.795.650	0,00%	1.838.053	1,80%
R14,F10,C8	2874660	2997290	2.997.290	0,00%	3.075.418	2,04%
R15,F01,C1	1042790	1039440	1.032.640	-0,66%	1.045.135	0,89%
R15,F05,C1	2297560	2170310	2.082.990	-4,19%	2.127.448	1,44%
R15,F10,C1	3304180	3194270	3.116.770	-2,49%	3.180.830	1,27%
R15,F01,C2	1176860	1205790	1.191.440	-1,20%	1.202.992	0,60%
R15,F05,C2	2723740	2698680	2.698.680	0,00%	2.725.057	1,02%
R15,F10,C2	4349910	4447950	4.310.340	-3,19%	4.397.752	1,26%
R15,F01,C8	2402800	2472860	2.465.650	-0,29%	2.473.680	0,44%
R15,F05,C8	5807050	6067350	5.969.370	-1,64%	6.044.925	1,06%
R15,F10,C8	9169890	10263600	9.304.650	-10,31%	9.840.033	4,61%
R16,F01,C1	140082	142692	140.149	-1,81%	141.531	0,70%
R16,F05,C1	259840	261775	261.775	0,00%	263.875	0,98%
R16,F10,C1	364786	374819	360.884	-3,86%	366.544	1,59%
R16,F01,C2	142381	145266	143.921	-0,93%	145.350	0,92%
R16,F05,C2	275626	277307	273.024	-1,57%	276.821	1,20%
R16,F10,C2	396966	391386	387.601	-0,98%	391.869	0,86%
R16,F01,C8	180199	187176	185.397	-0,96%	189.834	1,83%
R16,F05,C8	396721	423320	419.945	-0,80%	429.169	1,73%
R16,F10,C8	637944	649121	647.212	-0,29%	662.796	1,99%
R17,F01,C1	364784	374016	365.913	-2,21%	368.587	0,81%
R17,F05,C1	730195	718135	702.957	-2,16%	717.800	1,40%
R17,F10,C1	1150630	1041450	1.026.040	-1,50%	1.036.430	0,73%
R17,F01,C2	382593	393608	389.249	-1,12%	392.201	0,42%
R17,F05,C2	761041	786198	786.198	0,00%	800.117	1,23%
R17,F10,C2	1195710	1162290	1.159.440	-0,25%	1.172.777	1,08%
R17,F01,C8	531791	539817	539.817	0,00%	549.197	1,14%
R17,F05,C8	1284720	1348750	1.323.330	-1,92%	1.345.490	1,00%
R17,F10,C8	2047390	2227780	2.207.590	-0,91%	2.246.002	2,03%
R18,F01,C1	869263	864425	864.425	0,00%	879.355	0,88%
R18,F05,C1	1869230	1640200	1.627.700	-0,77%	1.668.745	1,73%
R18,F10,C1	2390740	2399230	2.366.280	-1,39%	2.425.505	2,95%
R18,F01,C2	980178	962402	962.402	0,00%	971.682	0,78%
R18,F05,C2	2146670	1958160	1.958.160	0,00%	1.987.065	1,59%
R18,F10,C2	n/a	2986000	2.986.000	0,00%	3.012.392	0,90%
R18,F01,C8	1560790	1617320	1.613.790	-0,22%	1.619.775	0,35%
R18,F05,C8	4230970	4268580	4.268.580	0,00%	4.421.490	2,75%
R18,F10,C8	n/a	7440780	7.194.120	-3,43%	7.610.975	4,53%

TABLE 9

6. Conclusion and Future research

In this paper we proposed scheduled transportation service network design model where the use and balance of vehicles is addressed. Traditionally, the allocation of vehicles to perform services has been addressed at a lower level and not when designing the service network. Thus the model presented here proposes a new perspective in planning transportation service networks.

The motivation behind the model comes from the fact that a significant cost factor in transportation networks arises in empty vehicle repositioning. The model here resolves this problem by adding a precedence constraint stating that a service may not be offered from a terminal unless a vehicle has arrived there on a preceding service. Assuming each possible service is operated by a vehicle, a constraint stating that the number of vehicles arriving at a terminal must equal to the number of services departing, can be added. This constraint will ensure that no service is offered unless a vehicle is available.

The mathematical formulation of the new constraints resembles flow balance constraints, except that they are forced upon the design variables. The constraints are thus dubbed design balance constraints. The design balance constraints impose a restriction on the arc selection by requiring an equal number of selected arcs entering and leaving each node in the network. If the design balance constraints are removed from the model the fixed-charge capacitated multi-commodity network design model formulation is obtained. Therefore we denote the new model with the addition of the design balance constraints the design balanced capacitated multi-commodity network design model (DBCMND).

Network design models are generally hard to solve. Adding the design balance constraints interconnects the binary decision variables and although this is not formally proved, computational results and intuition point to this problem being even harder. We therefore propose a Tabu Search heuristic framework to solve the DBCMND. The algorithm is based on two local search phases. The first phase, called the exploration phase, is based on a hybrid of a simple add/drop procedure and cycle based neighbourhoods, and is used to explore the solution space. The phase searches a neighbourhood of infeasible solutions. The second phase, called the feasibility phase, is added to explicitly find good feasible solutions.

Computational results show that the Tabu Search algorithm performed well on 78 network design instances with varying network attributes. Prior to the computational experiments, the algorithm's parameters were tuned to give the overall best performance regardless of network attributes. Comparing the results obtained from the algorithm with results from the Xpress-MP MIP-solver showed that the algorithm performed better for the largest selected network instances. For the very largest instances, where the MIP-solver did not find a solution within a 3600s time limit, the algorithm managed in all cases.

The good performance of the algorithm was further enhanced by simulating a simple parallel implementation with 6 different parameter configurations. Although some improvements were seen from alternating parameter settings, the main result from the parallel implementation is that the spreads of the results with the 6 parameter configurations are small. The low spread supports evidence that not only does the algorithm perform well, but it also is robust with respect to both parameter settings and network characteristics.

Although the computational results show the algorithm's good performance and robustness with respect to parameter settings and network attributes, there are a number of areas of improvements that should be considered in future research. The most evident is to take advantage of the cycle structure of design arcs in feasible solutions as made evident by the cycle based formulation of the DBCMND. One may consider investigating the possibility of solving the DBCMND using a column generation approach. Another

approach would be to include the cycle-based structure more evidently in the structure of the local search neighbourhoods. E.g. the two-phased approach might be replaced with a single local search phase, where an add/drop procedure (or any hybrid of it) could be used on cycles of arcs; thereby moving between feasible solutions. The difficulty in adopting this approach of more complex neighbourhood moves is the estimation of potential neighbour solutions without having to solve the CMCF problem to find the flow distribution on a given network design.

Considering the framework used for the algorithm presented here a possible area where to improve algorithm performance is using dynamic adjustment of the parameter configuration. The dynamic adjustment of the parameters could allow the algorithm to adapt itself to the characteristics of the instance it is applied to, eliminate superfluous computations, and thus improve performance. Another possibly more important area is to improve the guarantee of finding feasible solutions. Although the algorithm found feasible solutions to all 78 tested instances, it struggles on network instances with a sparse arc structure. This is due to the poor performance of the feasibility phase, where paths of arcs to open are hard to find, and paths of arcs to close often result in an infeasible CMCF problem. We expect the impact of the problem to be limited considering that real life service network design instances will be non-sparse. However, the problem could be addressed by introducing a third local search phase designed particularly for such difficult instances. The potential 3rd phase should, instead of using neighbourhood moves with full paths, use partial paths. This approach would diversify the imbalance structure by shifting the imbalance between nodes. This potential 3rd phase should be executed when the feasibility phase is unable to find a feasible neighbourhood solution.

Although there might be room for significant improvement or even alternative solution methods, the computational results presented here are conclusive enough to prove that the TS framework presented is a reasonable approach to solve the DBCMND problem. We believe the TS algorithm is powerful enough to allow the possibility of addressing vehicle balance requirements already at a service network design planning level; an opportunity that could significantly reduce the need for empty vehicle repositioning at an operational level.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Ahuja, R.K., Magnanti, T.J., Orlin, J.B., and Reddy, M.R. 1995. Applications of network optimization. In Ball, M., Magnanti, T.L., Monna, C.L., and Nemhauser, G.L., editors, Network models, volume 7 of Handbooks in Operations Research and Management Science, pages 1-83. North-Holland, Amsterdam.

Armacost, A.P., Barnhart, C., and Ware, K.A. 2002. Composite variable formulations for the express shipment service network design. Transportation science 36(1):1-20.

Balakrishnan, A., Magnanti, T.L., and Mirchandani, P. 1997. Network Design. In Dell'Amico, M., Maffoli, F., and Martello, S., editors, Annotated bibliographies in combinatorial optimization, pages 311-334. John Wiley & Sons, New York, NY.

Barnhart, C., Boland, N.L., Clarke, L.W., Johnson, E.L., Nemhauser, G.L., and Shenoi, R.G. 1998. Flight string models for aircraft fleeting and routing. Transportation Science 32(3):208-219.

Barnhart, C. and Kim, D. 1995. Routing models and solution procedures for regional less-than-truckload operations. Annals of operations research 61:67-90.

Cheung, R.K. and Muralidharan, B. 2000. Dynamic routing for priority shipments in LTL Service networks. Transportation Science 34(1):86-98.

Clarke, L., Johnson, E., Nemhauser, G. and Zhu, Z. 1997. The aircraft rotation problem. Annals of operational research 69:33-46.

Cordeau, J.-F., Toth, P., and Vigo, D. 1998. A survey of optimization models for train routing and scheduling. Transportation Science 32(4):380-404.

Crainic, T.G. 2000. Service network design in freight transportation. European journal of operational research 122:272-288.

Crainic, T.G., Gendreau, M., Soriano, P. and Toulouse, M. 1993. A tabu search procedure for multicommodity location/allocation with balancing requirements. Annals of operations research 41:359-383.

Crainic, T.G., Gendreau, M., Farvolden, J.M. 2000. A simplex-based tabu search method for capacitated network design. INFORMS journal on computing 12(3):223–236.

Dejax, P.J. and Crainic, T.G. 1987) A review of empty flows and fleet management models in freight transportation. Transportation Science 21(4):227-247.

Gamlouche, I., Crainic, T.G., Gendreau, M. 2003. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. Operations Research 51(4):655-667.

Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. Computers & operations research 1(3):533-549.

Glover, F. and Laguna, M. 1997. Tabu Search. Kluwer, Norwell, MA.

Jørgensen, R.M. and Madsen, O.B.G. 2004. Shortest path and real road networks. Paper under review. Developed at the Centre for Traffic and Transport, Technical university of Denmark.

Jansen, B., Swinkels, P.C.J., Teeuwen, G.J.A., van Antwerpen de Fluiter, B., and Fleuren, H.A. 2004. Operational planning of a large-scale multi-modal transportation system. European journal of operational research 156:41-53.

Joborn, M., Crainic, T.G., Gendreau, M., Holmberg, K., Lundgren, J.T. 2004. Economies of scale in empty freight car distribution in scheduled railways. Transportation Science 38(2):121-134.

Kim, D., Barnhart, C., Ware, K., and Reinhart, G. 1999. Multimodal express package delivery: A service network design application. Transportation Science 33(4):391-407.

Koskosidis, Y.A., Powell W.B., and Solomon, M.M. 1992. An optimization-based heuristic for vehicle routing and scheduling with soft time windows constraints. Transportation Science 26:69–85.

Kuby, M.J. and Gray, R.G. 1993. The hub network design problem with stopovers and feeders: the case of federal express. Transportation Research, Part A 27(1):1-12.

Magnanti, T.L., and Wong, R.T. 1984. Network design and transportation planning: models and algorithms. Transportation Science 18(1):1-55.

Marin, A. and Salmeron, J. 1996. Tactical design of rail freight networks. Part I: Exact and heuristic methods. European journal of operational research 90:26-44.

Minoux, M 1986. Network synthesis and optimum network design problems: models, solution methods and applications. Networks, 19:313-360.

Montgomery, D.C. 2000. Design and analysis of experiments, 5th edition. Wiley.

Newman, A.M. and Yano, C.A. 2000. Scheduling direct and indirect trains and containers in an intermodal setting. Transportation Science 34(3):256-270.

Nozick, L.K. and Morlok, E.K. 1997. A model for medium-term operations planning in intermodal rail-truck service. Transportation Research, Part A 31(2):91-107.

Pedersen, M. B. 2006. Optimization models and solution methods for intermodal transportation. PhD thesis, Centre for Traffic and Transport, Technical University of Denmark.

Powell, W. B. 1986. A local improvement heuristic for the design of less-than-truckload motor carrier networks. Transportation Science 20(4):246–357.

Powell, W.B. and Sheffi, Y. 1989. Design and implementation of an interactive optimization system for network design in the motor carrier industry. Operations research 37(1):12-29.

Appendix A

In order to find the most effective configuration of parameters for the TS algorithm, some initial experimentation is conducted. Ten different instances have been picked out from both the R and C data sets on which to carry out the parameter tuning experimentation. The ten instances are presented in table 1 in the paper's main sections. The data sets have been picked in different sizes and with different characteristic. The 'capacity ratio' is a ratio of commodity demand over total network capacity. The 'cost ratio' is a ratio of the fixed cost over the variable cost of arcs. The 'Opt. solution' column indicates whether an optimal solution was found for the instance solving the problem within a 3600s time limit using Xpress-MP's MIP solver.

There are eight parameters for which values have to be defined in the implementation of the Tabu Search framework we have presented. To determine appropriate values for these eight parameters we apply a 2-level 2^8 factorial experimentation plan [Montgomery 2000] where each of the eight parameters is evaluated on two levels, a low level and a high level. The two levels chosen for each of the eight parameters are:

- Iteration range l_i, 10 and 30
- Percentage gap l_g, 0.5% and 5%
- Penalty scaling factor ψ^p, 0.5 and 2
- Length of the Tabu list l_t, 5 and 25
- Number of fixed cost candidates l_f, 5 and 15
- Number of variable cost candidates l_v, 5 and 15
- Number of residual capacity candidates l_r, 5 and 15
- Number of penalty value candidates l_p, 5 and 15

The two levels of each parameter are determined from intuitive considerations. All of the levels have been determined at two extremities from the consideration that their effects would be best tested with extreme values. However the extremities are chosen with moderation as each of the parameters intuitively will have no effect if set to 0 and either produce poor results or have no effect at very high levels.

Aggregated computational results for the 32 runs (600s.) on selected instances for parameter tuning								
instance name	Xp. Sol	Bound	Sol. Gap	Time (s)	TS. Avg. TS. Best	Avg. Xp. GAP Best Xp. Gap	Avg. Bound Gap Best Bound Gap	Spread (% of Avg.)
R10,F05,C2	443.149	443.149	0,00%	1198	449.705 443.547	1,46% 0,09%	1,46% 0,09%	1,02%
R12,F10,C2	7.408.996	7.408.996	0,00%	1055	7.760.968 7.530.870	4,54% 1,62%	4,54% 1,62%	1,66%
R13,F01,C8	218.787	218.787	0,00%	2332	226.181 223.231	3,27% 1,99%	3,27% 1,99%	0,76%
R15,F10,C8	9.105.014	9.105.014	0,00%	1131	9.745.193 9.366.760	6,57% 2,79%	6,57% 2,79%	2,64%
C20,230,200,F,L	160.923	128.014	20,45%	(t)	150.180 146.643	-7,15% -9,74%	14,76% 12,70%	1,90%
R16,F10,C1	379.910	309.383	18,56%	(t)	362.252 352.681	-4,87% -7,72%	14,59% 12,28%	2,17%
R17,F01,C1	364.784	364.784	0,00%	1895	369.441 365.801	1,26% 0,28%	1,26% 0,28%	1,05%
R18,F05,C1	1.986.164	1.362.596	31,40%	(t)	1.673.958 1.597.610	-18,65% -24,32%	18,60% 14,71%	2,25%
C30,520,100,V,T	53.219	52.662	1,05%	(t)	54.569 53.972	2,47% 1,40%	3,49% 2,43%	0,56%
C100,400,30,F,L	81.638	58.316	28,57%	(t)	75.941 67.603	-7,50% -20,76%	23,21% 13,74%	5,18%

TABLE 10

The total length of the candidate list is determined from the size of each of the four candidate sub-lists. It is estimated that no less that 20 candidates should be tested considering the network sizes of the instances of often more than 100 arcs. Accrediting each sub-list equal importance the low level for each of the sub-list parameters l_f, l_v, l_r, and l_p is set to 5. The high level of 15 is chosen as an estimated significant difference. The tabu list was set at a low level of 5 in comparison to the minimum size of the candidate list of 20. The high level of 25 is estimated to be a reasonable high extremity. The iteration range levels were chosen according to the Tabu list length levels with the low level staying below the high level of the Tabu tenure and the high level above the high level Tabu tenure. The improvement gap percentage was chosen with a high level of 5% and a low tight level of 0.5%. Finally the penalty scale factor is chosen to test the importance of the penalty value at half or double value.

Testing all the combinations of parameters at both levels require 2^8 experiments for each of the ten instances. To conduct the $10 \cdot 2^8$ experiments significant computational time would be needed. To limit computational requirements we reduce the experimentation plan to a fractional 2^5 factorial design by confounding 3 of the parameters with higher degree interactions. This is reasonable to do if it is assumed that parameter interaction is limited. A total of 32 experimentation runs are performed using the TS algorithm for the selected instances for the parameter tuning. We also limit the computational time to 600 seconds. This further reduces the computational effort required to perform the experimentation plan, while still giving a reasonable indication of the search process. It is important to note here that the reason why

we allow ourselves to differentiate between the run time for the MIP-solver (3600 s.) and the TS-algorithm run time (600 s.) is that no significant comparisons between the MIP-solver and the TS-algorithm are done at this time. The focus of the computations is to evaluate the impact from each of the parameters within the algorithm itself. The MIP-solutions are only added to provide a benchmarking point by differentiating between the problems solved to optimality by the MIP-solver within a 3600s time limit, and those not.

The results from the parameter tuning experiments are presented in table 10. The first four columns show the instance name, the best found solution using the Xpress-MP MIP solver within a time limit of 3600s., the lower bound obtained from the Xpress-MP solution, and the relative gap between the best obtained solution and the lower bound. The fifth column show the solution time to find the optimal solution or a (t) if the solver was stopped after 3600s. The following column shows, in pairs, the average solution value of all 32 runs on the instance and the best found solution value using the TS-algorithm. The following column shows, also in pairs, the relative gap between the best MIP-solver solution and the average solution and best solution found with the TS-algorithm. The second last column shows, once again in pairs, the gap between the lower bound obtained from the MIP-solver, and the average solution and best solution found with the TS-algorithm respectively. The last column shows the spread of the 32 solutions relative to the average solution value.

The most important observation to be taken from the initial parameter tuning is the spread of the results. For all instances, except instance C100,400,30,F,L, the spread of the solution values is less than 3% from the best average solution. This indicates that the algorithm is robust towards parameter selection. This is an important attribute for practical application of the algorithm, and significantly reduces the importance of the effort of parameter value selection. The exception of instance C100,400,30,F,L having a higher spread is presumably not caused by the different parameter value settings. Rather it was observed that cycling between the solutions in the two local search phases was observed along with a high degree of failure to find a feasible solution in the feasibility phase. The later is due to the sparseness of the graph leading to fewer feasible candidate solutions. A careful conclusion would be that the algorithm is not suitable for sparse networks. It would therefore appear that a diversification method to avoid cycling and/or an enhanced feasibility phase to improve the search for feasible solutions would be beneficial.

The effects of the eight parameters in the 2^5 experiments are calculated using Yates' algorithm. The results from using Yates' are used to get an indication of the best parameter setting. Because of the robustness of the algorithm with respect to the parameter configurations, the evaluation of the effects has not been subject to statistical analysis. Their significance has been interpreted subjectively. Table 11 shows the optimal configuration of the parameter values for each of the ten instances based on the 32 runs performed with the TS-algorithm. When the table reads *"low"* it indicates that the best results are achieved with the parameter at its low level, while *"high"* indicates the opposite. The words put in parenthesis indicate that only a relatively small effect is registered from the effect of the parameter. Where nothing is written the value of the parameter has no effect in the tested range. If all the effects from each of the ten instances are aggregated into one "optimal" parameter configuration we achieve the configuration illustrated at the bottom of the table 11.

The setting of the fixed cost sub-list length parameter is somewhat inconclusive. The optimal configuration indicates that high values should be applied for the improvement gap, the tabu list length, the variable cost sub-list length and the residual capacity sub-list length while low values should be applied to the penalty value sub-list length, the penalty scaling factor and the iteration range. An intuitive reason for having a high level for the improvement gap combined with a low level for the iteration range

is that the feasibility phase is executed more often and thus more feasible solutions are obtained. The reason for the high tabu tenure being better than the short one could be accredited to the fact that keeping arcs tabu diversifies the search more. Furthermore the combination between having longer tabu tenure than iteration range allows the tabu tenure memory to survive between two consecutive exploration phase executions and thereby prevent cycling between solutions.

	Best parameter selection from initial computational runs							
instance name	Parameter							
	# f. cost	# p. cost	# residual	#v. cost	tabu ten.	p. scale	imp. gap	it. range
R1O,F05,C2	low	low	high			low	high	low
R12,F1O,C2		low						low
R13,FO1,C8				(low)		(low)		(low)
R15,F1O,C8		(low)				(low)		low
C20,230,200,F,L				(high)		(low)		
R16,F1O,C1	low		high	(high)	high	low		low
R17,FO1,CI	high		high	high	high		(high)	
R18,F05,C2								
C30,520,1OO,V,T								
C1OO,400,30,F,L,1O	high	low	high		high			low
	Aggregated parameter selection indication							
		low	high	high	high	low	high	low

Table 11